VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Nástroj pro vytváření myšlenkových map ve OneNote

Mind Map Tool for OneNote

Daniel Harbáček

Bakalářská práce

Vedoucí práce: Ing. Svatopluk Štolfa, Ph.D.

Ostrava, 2021

Abstrakt

Tato bakalářská práce pojednává o myšlenkových mapách a vytvoření doplňku, který usnadňuje jejich vytváření v aplikaci OneNote pro web. V teoretické části práce je uvedena specifikace Microsoft Office a myšlenkových map, včetně principu jejich tvorby a představení stávajících doplňků pro aplikaci OneNote. Dále jsou v této části popsány techniky a postupy, jež byly použity při vytváření takového doplňku. Druhá část práce je věnována samotné implementaci, která zahrnuje návrh doplňku, jeho architekturu a funkce. Výstupem práce je tedy funkční a uživatelsky přívětivý doplněk usnadňující vytváření myšlenkových map v aplikaci OneNote pro web.

Klíčová slova

Microsoft Office; OneNote pro web; Office doplněk; Myšlenková mapa; Office API; JavaScript; HTML; CSS

Abstract

This bachelor thesis is about mind maps and the creation of an add-on that simplifies their creation in the OneNote application for the web. The theoretical part of the thesis presents the specifications of Microsoft Office and mind maps, including the principle of their creation and introduction existing add-ons for OneNote. The techniques used to create such a supplement are further described in this section. The second part of the work is devoted to the implementation itself, which includes the architecture of the add-on and its functions. The output of the work is a functional and user-friendly add-on that simplifies the creation of mind maps in the OneNote application for the web.

Keywords

Microsoft Office; OneNote for the web; Office add-on; Mind Map; Office API; JavaScript; HTML; CSS

Poděkování

Rád bych poděkoval Ing. Svatopluku Štolfovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Obsah

Seznam použitých symbolů a zkratek

Zkratka	Význam
API	Application Programming Interface
COM	Component Object Model
CSS	Cascading Style Sheet
ECMA	European Computer Manufacturers Association
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
OS	Operační systém
UML	Unified Modeling Language
URL	Uniform Resource Locator
VBA	Visual Basic for Applications
VSTO	Visual Studio Tools for Office
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V dnešní digitální době, která se nese ve znamení především distanční výuky, jsou myšlenkové mapy vhodnou pomůckou či nástrojem jak pro žáky a studenty, tak i pro učitele. Jejich úkolem je poskytnout studentům a žákům ucelený přehled nad danou problematikou, a tím jim ulehčit náročnou formu distanční výuky. Profesoři a učitelé je mohou použít jako podklad pro probíranou látku či vysvětlení složitějších témat. Škola však není jediným prostředím, kde lze tyto mapy využívat. Své místo nalezly i v činnostech každodenního života, jako je organizování času, obyčejného plánování, nebo v oblasti firemního vzdělávání, růstu a při řízení projektů.

Klasickou formu myšlenkových map známe jako na papíře nakreslené grafické znázornění analyzovaného problému, který by byl prostým zápisem ve formě slov špatně pochopitelný. Nyní již existuje spousta možností, jak myšlenkové mapy znázorňovat i bez papíru. Slouží k tomu celá řada pokročilých prostředků v podobě různých programů, aplikací nebo nástrojů integrovaných v prohlížeči.

Jednoduše lze myšlenkové mapy charakterizovat jako specifický náčrt složený z různých tvarů či obrazců, jež představují dané problémy, které chce autor analyzovat. Ty dále větví pomocí spojnicových čar do podproblémů a snaží se tím dané téma zjednodušit. Je vhodné pro vytváření takovýchto map zapojit fantazii. Čím více kreativity, barev a kreseb autor zapojí, tím je téma atraktivnější, lépe zapamatovatelné a víc se odlišuje od obyčejného zápisu, který není tak konstruktivní. [1]

Cílem práce je vytvoření uživatelsky přívětivého a funkčně zdatného doplňku aplikace OneNote pro web v rámci které dojde k zefektivnění vytváření myšlenkových map.

Původně měl být doplněk vytvářen pro OneNote verzi 2016, avšak po konzultaci s vedoucím této bakalářské práce a vzájemné domluvě bylo od tohoto záměru upuštěno a doplněk byl vyvíjen pro webovou verzi aplikace OneNote (dále jen „aplikace OneNote“).

Důvodů, proč bylo toto rozhodnutí provedeno, je více. Do aplikace OneNote ve verzi 2016 pro Windows lze přidávat pouze doplňky typu COM, což je zastaralý způsob rozšiřování Office aplikací, který je k dispozici již od verze Office 2000. Pomocí takto vytvořených doplňků lze do OneNote aplikace pro operační systém Windows přidat prvky uživatelského rozhraní, jako jsou tlačítka v panelu příkazů či nový pás karet. Doplněk vytvořený pomocí technologie COM však neposkytuje všechny prvky uživatelského rozhraní a možnosti rozšíření OneNote aplikace pro Windows, které potřebujeme k vytvoření doplňku pro vytváření myšlenkových map. V roce 2016 Microsoft představil nově vytvořené API, jež je určeno k interakci s OneNote aplikací a které představuje moderní způsob

vytváření doplňků pomocí webových technologií. Jelikož Microsoft společně s balíkem Office 2019 nevydal i nové verze aplikací OneNote tak, jak to bývalo zvykem, je jedinou aplikací, která podporuje toto moderní API OneNote pro web. Na rozdíl od některých dalších Office aplikací se v dřívějších verzích aplikace OneNote pro Windows nenachází ani zpětná podpora tohoto API. [2] [3] [4]

Druhý důvod je ten, že se tato aplikace nachází ve fázi všeobecné podpory pouze do 10. října 2023. To znamená, že do něj budou přidávány nové funkce už jen po velmi krátkou dobu. Po tomto datu OneNote 2016 (nyní přejmenován na OneNote) přejde do fáze takzvané rozšířené podpory, která potrvá do 14. října 2025. V tomto životním cyklu již do aplikace nebudou přidávány žádné nové funkce a budou pro ni vycházet jen opravy. [5]

Tato bakalářská práce bude zahrnovat průzkum aktuálně dostupných možností online tvorby myšlenkových map, které lze integrovat do aplikace OneNote. Dále bude obsahovat přehled a specifikace jednotlivých technologií, které budou použity pro tvorbu doplňku, jež bude usnadňovat vytváření, úpravu a možnosti sdílení myšlenkových map v rámci OneNote pro web. Nakonec bude uveden popis klíčových prvků, ze kterých se doplněk skládá a budou nastíněny možnosti, jak doplněk dále rozšířit.

Kapitola 2

Specifikace Microsoft Office a myšlenkových map

Tato kapitola pojednává o programu používaném k vytváření myšlenkových map, o tom, jak jej získat společně s doplňky, jejich výhody a nevýhody a bude vysvětleno, jak se tyto doplňky používají. Následně budou popsány myšlenkové mapy, k čemu se používají, jejich vznik a užitečné postupy při jejich tvorbě.

2.1 Microsoft 365

Microsoft 365 reprezentuje kancelářský balík firmy Microsoft dostupný pro operační systémy Windows i Mac OS. Obsahuje širokou škálu aplikací a služeb s různým využitím jak pro jednotlivce nebo studenty, tak pro firmy a podnikatele. Microsoft tento balík vydává ve verzích, přičemž poslední, nejnovější verze vyšla 24. září 2018. Microsoft tento balík nabízí v několika edicích. Každá edice je určena pro jinou cílovou skupinu osob, jinému využití a zahrnuje jiné aplikace či služby (viz. tabulka 2.1). [6]

Tabulka 2.1: Porovnání edicí Office 365 [7], vlastní zpracování

Pro rodiny	Pro studenty a domácnosti	Pro jednotlivce	Business Basic	Business Standard	Business Premium	Apps
Word	Word	Word	Exchange	Outlook	Outlook	Outlook
Excel	Excel	Excel	OneDrive	OneDrive	OneDrive	OneDrive
PowerPoint	PowerPoint	PowerPoint	SharePoint	Word	Word	Word
OneNote		OneNote	Microsoft Teams	Excel	Excel	Excel
Outlook		Outlook		PowerPoint	PowerPoint	PowerPoint
OneDrive		OneDrive		SharePoint	SharePoint	Publisher
Family Safety		Skype		Microsoft Teams	Microsoft Teams	Access
Skype				Exchange	Exchange	
				Publisher	Publisher	
				Access	Access	

Stěžejní aplikací pro tuto práci je Microsoft OneNote, který zde bude podrobněji vysvětlen. Každý kancelářský balík, který je k dispozici, poskytuje přístup k aplikaci OneNote, a to jak ve verzi pro web, tak pro operační systémy Windows a macOS. Jeho používání na různých platformách, včetně webové aplikace, je zdarma avšak úložný prostor, který aplikace využívá, je omezen. Uživatelé, kteří si nepředplatí žádný balík Office 365, mají k dispozici pouze 5 GB úložného prostoru pro jejich poznámkové bloky. Studenti využívající edici OneNote for Education, mohou využít až 1 TB místa na OneDrive pro jejich poznámky. Po jeho naplnění je třeba zakoupit některý z výše uvedených balíků, který zahrnuje OneNote. [8]

Microsoft OneNote představuje aplikaci, která je součástí balíku Microsoft Office a slouží k ukládání jakýchkoliv poznámek, myšlenek či nápadů, přičemž do jednoho poznámkového bloku může přispívat větší počet uživatelů. Uživatelé, kteří vlastní licenci pro používání tohoto programu, mají k dispozici takzvané poznámkové bloky, které se významem shodují s klasickým poznámkovým blokem z reálného světa. Každý poznámkový blok je rozdělen na sekce. Každá z těchto sekcí obsahuje stránky, které uživateli slouží k sepsání poznámek nebo nápadů. Dále OneNote umožňuje kreslení, vkládání obrázků, nahrávání zvukových poznámek, vložit online videa, přidávat soubory a mnoho jiného. Všechny uživatelem vytvořené poznámky se automaticky ukládají na cloud, takže jsou k dispozici z jakéhokoliv zařízení. OneNote také umožňuje sdílet poznámkové bloky s ostatními uživateli, což znamená, že v jednu chvíli může jednu stránku upravovat více lidí současně.

2.2 Myšlenkové mapy a jejich vznik

Myšlenková mapa, někdy také označována jako mentální mapa, je nástroj, který umožňuje zachycení myšlenkových toků do grafické podoby. Je velmi oblíbenou pomůckou, jelikož dokáže velmi přesně zachytit procesy probíhající v mozku. Své využití nachází při plánování úkolů, řešení problémů, učení a při řešení spousty dalších každodenních činností. [1]

Za jejího vynálezce je označován Tony Buzan, anglický autor a psycholog, který v 60. letech 20. století přišel s myšlenkou, že zápisky nemusí být pouze úhledné a upravené, ale mnohem lépe se vstřebávají do paměti, když jsou barevné a různě poskládané do jakési stromové struktury. Vizuální a písemné znázornění myšlenek se spojí a vytvoří něco, co je přirozenější pro lidský mozek než jednoduché poznámky na papíře.

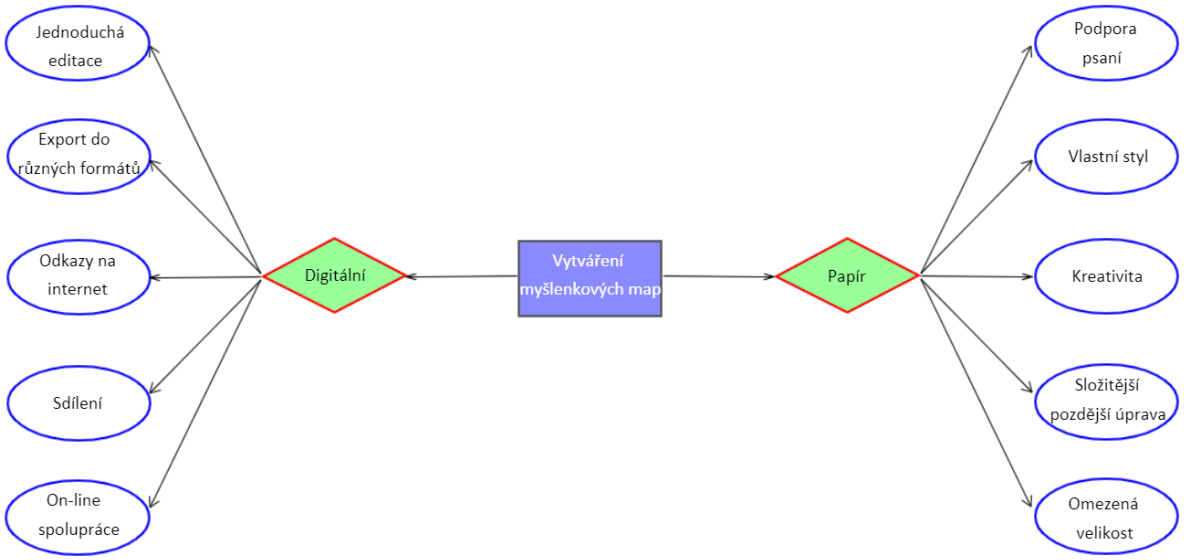
Původně myšlenkové mapy zamýšlel jako způsob, jak pomoct žákům a studentům vytvářet si poznámky pouze s klíčovými slovy a obrázky, ale v průběhu času se staly jakýmsi obecným formátem, který umožňuje zachytit jakékoliv nápady či zapisovat poznámky ve formě jednoduché k porozumění a zapamatování. Podle Buzana byly myšlenkové mapy využívány světovými génii napříč věky. Například zápisky Leonarda da Vinciho obsahovaly kresby se spojením slov a poznámek.

Lidský mozek má dvě hemisféry, levou a pravou. Levá hemisféra bývá popisována především jako logická, racionální, a jako ta, která udržuje věci organizované. Kdežto, pravou hemisféru můžeme chápat jako intuitivní a tvořivou. Tony Buzan uvádí, že tradiční forma psaní poznámek, zahrnující odstavce čistého textu, je špatně zapamatovatelná, protože mozek používá pouze levou, logickou hemisféru, zatímco pravá hemisféra není vůbec zapojena. Tradiční zapisování poznámek zleva doprava a seshora dolů není pro mozek tak přirozené, jako když vnímá celou stránku najednou jako celek. Buzan

tvrdí, že používání obrázků a tvořivosti při psaní poznámek proniká do mozku mnohem lépe, protože využívá obě hemisféry. [9] [10]

2.3 Princip tvorby myšlenkových map a stávající nástroje pro jejich vytváření

Myšlenková mapa se skládá z témat. Každé téma obsahuje klíčové slovo, případně menší počet slov, která ho jasně a výstižně popisují. Myšlenkovou mapu si můžeme představit jako strom. Uprostřed se nachází hlavní téma, které reprezentuje kmen a z něj vychází podtémata, která reprezentují větve, případně listy. Začneme tedy tím, že si doprostřed papíru napíšeme slovo, které určuje téma naší myšlenkové mapy. Kolem tohoto slova se obvykle nakreslí kruh, obdélník, či jakýkoliv jiný obrazec. Kreativnější lidé mohou kolem tématu nakreslit jakýkoliv obrázek, případně ho různě vybarvit. Fantazii se meze nekladou. Toto téma si označíme jako hlavní. Jakmile máme vytvořené hlavní téma, vytváříme postupně další témata tak, že vedeme šipku z jednoho tématu k druhému. Tento postup opakujeme, dokud nevyčerpáme všechna témata, která jsme chtěli napsat. Při tvoření bychom neměli zapomínat na kombinaci kreslení a psaní textu, protože tím zapojujeme obě hemisféry, díky čemuž bude pro mozek mnohem jednodušší vstřebat poskytnuté informace. Příklad vytvořené myšlenkové mapy pomocí doplňku v aplikaci OneNote je vyobrazen na obrázku 2.1.



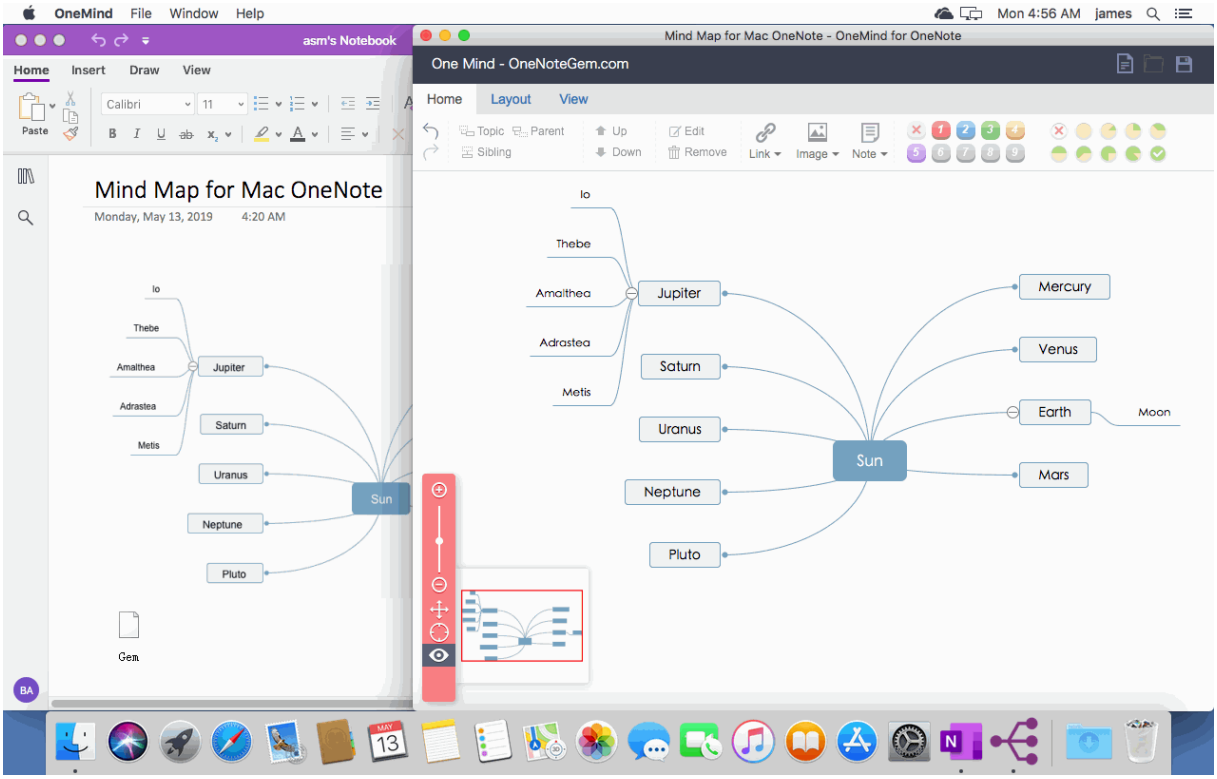
Obrázek 2.1: Příklad myšlenkové mapy

Aplikace OneNote umožňuje kreslení různých obrazců, tvarů a čar, které se mohou různě barevně kombinovat. Mimo jiné můžeme využít pera různých tlouštěk a barev, díky kterým vypadá výsledná myšlenková mapa mnohem kreativněji. OneNote také umožňuje přidat text na jakékoliv místo, přičemž ho lze barevně zvýrazňovat či měnit jeho velikost. Aplikace tedy obsahuje všechny prostředky, které jsou k vytváření takovýchto map potřebné. Jejich tvorba s využitím pouze základních nástrojů je však příliš zdlouhavá a při vytváření velké, rozlehlé mapy se její vytváření stává příliš těžkým úkolem. Uživatelům, kteří myšlenkové mapy nepoužívají pravidelně, sice bude pravděpodobně stačit pro jejich vytváření OneNote ve své základní formě. Ovšem uživatelé, kteří si myšlenkové mapy oblíbili a vytvářejí je opakovaně, dokážou ocenit doplněk, jehož cílem je jejich tvorbu zjednodušit. Myšlenková

mapa vytvořena pomocí takového nástroje se příliš neliší od myšlenkové mapy vytvořené bez něj. Rozdíl netkví ve výsledném vzhledu, ale v postupu její tvorby. Výsledná myšlenková mapa může být s použitím doplňku vytvořena až několikanásobně rychleji, než kdyby byla vytvářena s použitím pouze základních nástrojů. Výhoda takového doplňku tedy spočívá v jednoduchém, interaktivním a souvislém procesu tvorby, který se takto stává velice snadným úkolem, jež uživatele stojí minimum času.

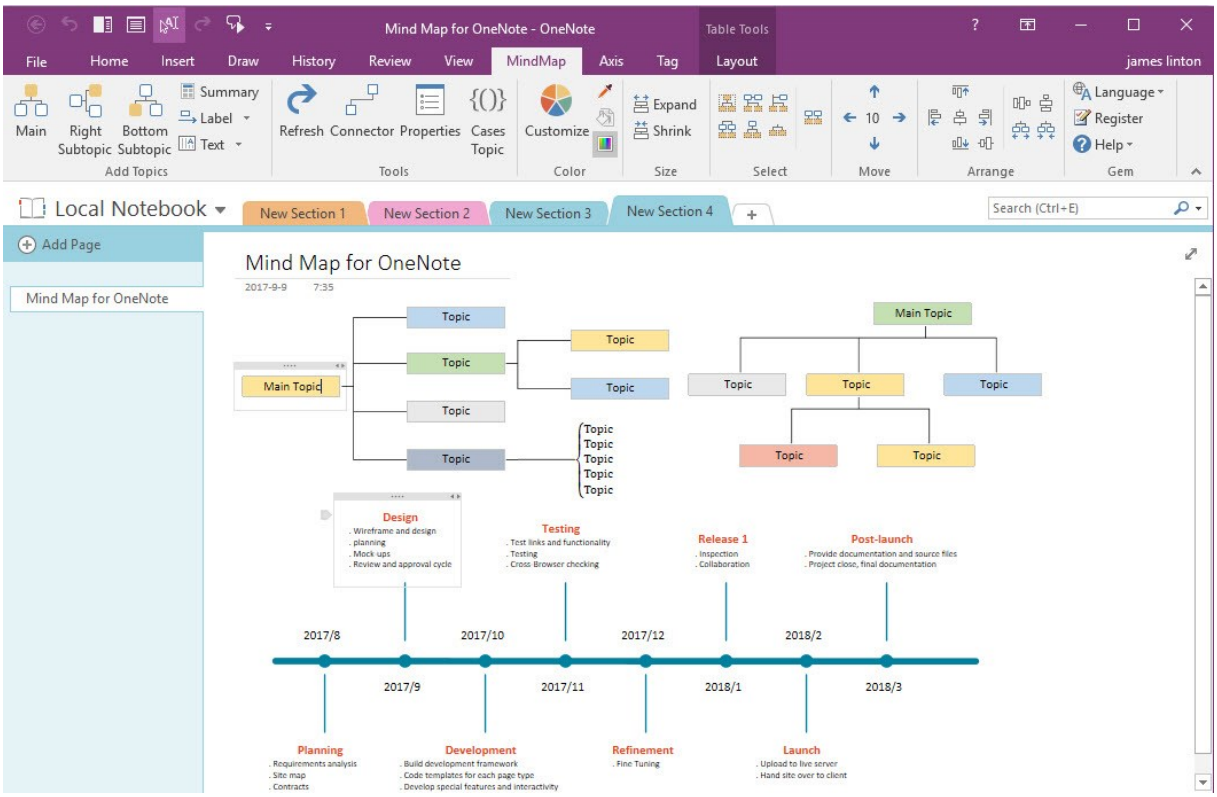
Pro vytváření myšlenkových map v aplikaci OneNote či jejich export do této aplikace již existují doplňky, které se o takovou funkcionalitu postarají. Dále bude popsáno, jak se některé z nich používají nebo jaké jsou jejich výhody a nevýhody.

OneMind představuje multiplatformní rozšíření pro OneNote od společnosti Digital GemSoft Ltd. Jeho přístup k řešení problému je trochu odlišný, než u jiných rozšíření či doplňků. OneMind se totiž instaluje jako samostatná desktopová aplikace, která umožňuje vytváření myšlenkové mapy a její následný export do OneNote pro operační systémy Windows a macOS ve formě obrázku. Přestože desktopová aplikace není ničím omezena, nabízí pouze základní funkce, jako je přidávání nových témat, změna jejich barvy či vložení obrázků. Jak již bylo zmíněno, poté, co uživatel vytvoří myšlenkovou mapu, ji může exportovat do OneNote ve formě obrázku pomocí jediného kliknutí. Tuhle myšlenkovou mapu však nelze nijak upravovat, takže při potřebě jakékoliv její editace je uživatel odkázán na desktopovou aplikaci. Mezi nevýhody tohoto software také patří to, že nepodporuje exportování myšlenkové mapy do OneNote 2019 či OneNote pro web.



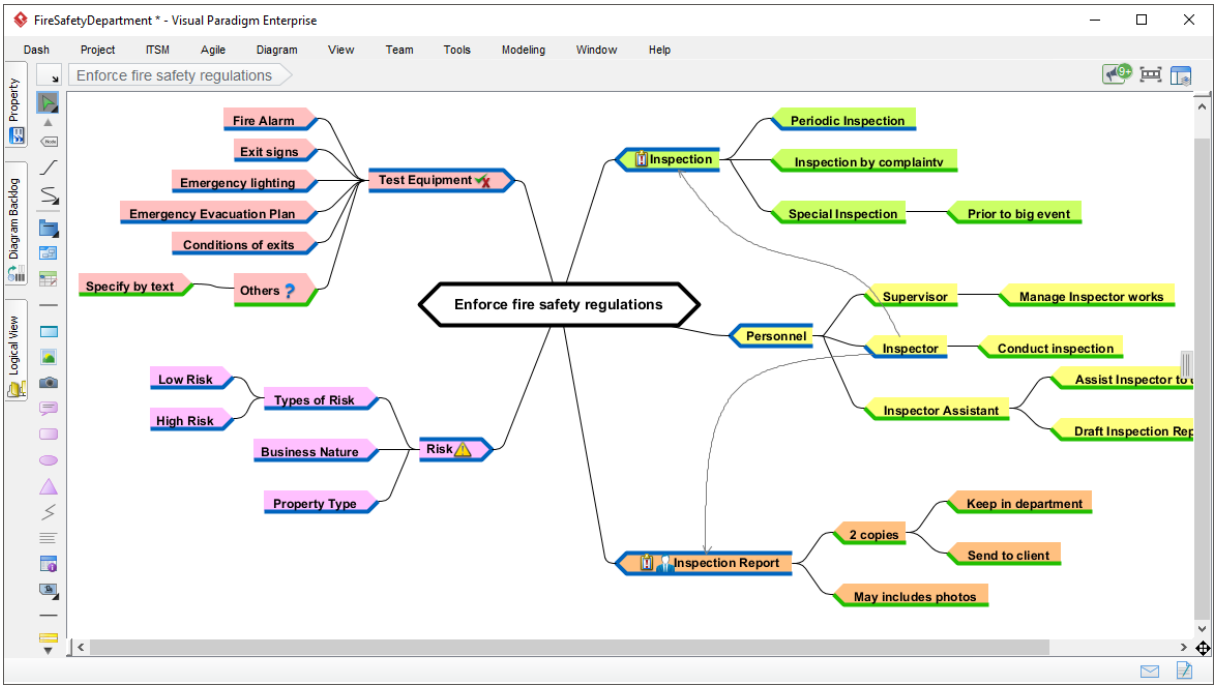
Obrázek 2.2: Tvorba myšlenkové mapy pomocí nástroje OneMind for OneNote [11]

MindMap for OneNote reprezentuje další doplněk pro tvorbu a editaci myšlenkových map vytvořený společností Digital GemSoft Ltd. Ten již přináší nové funkce přímo do OneNote pro Windows. S pomocí tohoto doplňku uživatel může vytvořit a editovat jednoduchou myšlenkovou mapu, která se zobrazuje ve formě nativních objektů. Oproti předchozímu doplňku zahrnuje pokročilejší funkce, jako například zmenšení či zvětšení vzdálenosti mezi jednotlivými tématy, pohyb s tématy nebo určení směru, ve kterém bude přidáno další téma. Pokud ovšem bude chtít uživatel pohybovat tématy manuálně, doplněk se o tom nedozví a musí kliknout na tlačítko, které myšlenkovou mapu překreslí. Další jeho nevýhodou je to, že je doplněk dostupný pouze pro OneNote 2010, 2013 a 2016 v operačním systému Windows.

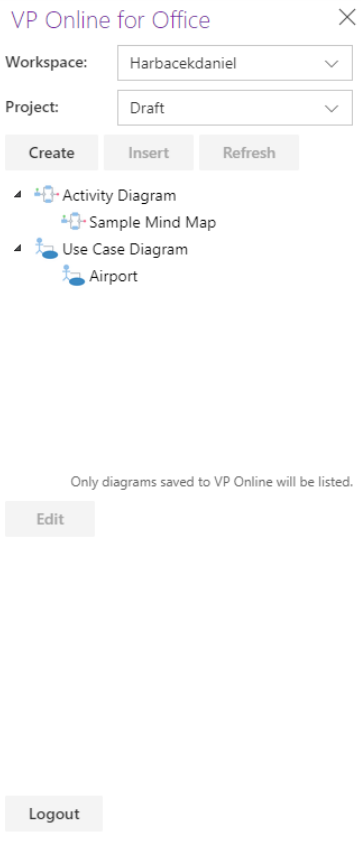


Obrázek 2.3: Tvorba myšlenkové mapy pomocí nástroje Mind Map for OneNote [12]

Visual Paradigm je nástroj pro tvorbu diagramů, schémat a grafů mnoha různých typů. Ty lze editovat jak v online prostředí webového prohlížeče, tak pomocí desktopové aplikace, kterou je nutno nainstalovat. Pomocí Visual Paradigm lze vytvořit velký počet UML diagramů, mezi které patří diagram aktivit či diagram tříd, dále také entito relační diagram, vývojové diagramy, ale i myšlenkové mapy a mnoho dalších. Pro jejich vytvoření ve webové či desktopové aplikaci stačí přetáhnout objekty ze záložky na stránku a spojit je čarou. Dále je možno jednotlivé prvky různě barevně zvýrazňovat. Interakce s aplikací OneNote je zajištěna pomocí doplňku, díky kterému může uživatel po předchozím přihlášení vložit jakýkoliv dříve nakreslený diagram do aplikace OneNote jako obrázek. Nevýhodou takového přístupu je ale to, že takhle vytvořenou myšlenkovou mapu nelze nijak upravovat.



Obrázek 2.4: Tvorba myšlenkové mapy pomocí nástroje Visual Paradigm [13]



Obrázek 2.5: Importování myšlenkové mapy pomocí doplňku Visual Paradigm do OneNote

Kapitola 3

Programovací jazyky a API používané k vývoji doplňků ve OneNote

V této kapitole si charakterizujeme programovací jazyky, jejich konstrukce a knihovny použité k vývoji doplňku aplikace OneNote. Dále bude popsán způsob vytváření doplňků, API, která byla použita ke komunikaci s OneNote a možnosti nasazení takto vytvořených doplňků.

3.1 JavaScript

JavaScript bývá popisován jako interpretovaný, multiplatformní skriptovací programovací jazyk, standardizován specifikací ECMA Script. Jelikož je interpretovaný, nedochází k jeho kompilaci a je prováděn tak, jak byl jeho kód napsán. Jeho běhovým prostředím je internetový prohlížeč na straně klienta, ale s použitím frameworku jako je Node.js lze JavaScript spustit i mimo prohlížeč. Používá se k popisu chování webových stránek při výskytu nějaké události. Jeho zdrojový kód se do stránky ukládá jako součást HTML kódu. Obvykle se používá ve spojitosti s jazyky HTML a CSS, které budou popsány dále. Často bývá mylně označován jako „interpretovaná Java“, či „jazyk vycházející z Javy“. Syntaxe JavaScriptu se sice velmi podobá Javě, ale tímto výčet podobných rysů končí. [14]

JavaScript má ve webovém prohlížeči mnoho využití. Může být použit například pro definici chování webové stránky nebo pro modifikaci jejího obsahu či vzhledu. K tomu využívá objekt document, který reprezentuje webovou stránku. Prostřednictvím tohoto objektu lze vytvářet, odebírat, či měnit HTML elementy a jejich vlastnosti, případně na nich zachytávat události. Mezi další možnosti použití JavaScriptu patří využívání API třetích stran, vývoj her běžících v prohlížeči, zobrazování animací a mnoho dalšího.

Proměnné se v JavaScriptu inicializují klíčovým slovem *var*. S příchodem ECMA Script 2015 (ES6) se tato možnost rozšířila o dvě nová klíčová slova: *let* a *const*. Proměnné inicializované pomocí *let* nebo *const* jsou platné pouze v rozsahu daného bloku, ve kterém jsou definovány, zatímco proměnná typu *var* existuje buď v globálním rozsahu, pokud je inicializována mimo funkci, nebo v rozsahu funkce, ve které je definována. Proměnné *let* a *const* také zavádí striktnější, korektnější pravidla použití podobná jazykům C# nebo Java. Oproti proměnné typu *var* je nemůžeme znovu deklarovat v rámci stejného rozsahu, avšak opětovná deklarace proměnné let nebo const je možná pouze pokud se nachází v jiném rozsahu či bloku. Přestože jsou tyto proměnné beztypové, můžeme

jim přiřadit hodnoty různých datových typů, například číslo, řetězec, nebo objekt. V následující tabulce jsou přehledně znázorněny hlavní rozdíly mezi těmito klíčovými slovy. [15]

Tabulka 3.1: Rozdíly mezi klíčovými slovy const, let a var, vlastní zpracování

Klíčové slovo	var	const	let
Opětovná deklarace	ANO	NE	ANO
Blokový rozsah	NE	ANO	ANO
Funkční rozsah	ANO	ANO	ANO
Patří objektu window	ANO	NE	NE

Syntaxe JavaScriptu, jak už bylo zmíněno, je příbuzná jazykům Java nebo C#. Mezi základní konstrukce jazyka patří podmínka *if*, rozhodovací blok *switch*, cykly *for*, *while* a *do-while* a chybový blok *try catch*. Tyto konstrukce se používají téměř stejně jako v jazycích C# nebo Java. Pole v JavaScriptu jsou speciální případ objektu. Hlavním rozdílem je to, že se pole indexují pomocí čísla namísto jména, ale je možno vytvořit i takové pole, které bude mít jako index text ve formě řetězce. Jednomu prvku v poli můžeme přiřadit proměnné různých typů, dokonce i objekty či funkce. Pole také poskytují velké množství užitečných funkcí, jako je procházení pole pomocí *forEach*, přidávání a odebírání prvního nebo posledního prvku, rozdělování či sjednocování polí. [16]

ECMA Script 2015 (ES6), který představil dvě klíčová slova const a let, uvedl také vytváření tříd. V dřívějších verzích se třídy nahrazovaly používáním funkcí a prototypů, jež tvořily objekty, kterým jsme mohli přiřadit vlastnosti i metody. Třída představuje v JavaScriptu šablonu pro vytváření objektů se společným chováním. Zapouzdřuje data, která spolu souvisí tak, aby byla uchovávána v jediné proměnné.

Deklarace třídy se provede klíčovým slovem *class* následovaným jménem třídy a do složených závorek se může uvést konstruktor, případně funkce, které bude třída obsahovat. Dále ES6 zavedl klíčové slovo *static*, které definuje statickou vlastnost nebo metodu dané třídy, kterou mají všechny její objekty společné. JavaScript také umožňuje definovat privátní vlastnosti třídy tak, že se na začátek jména vlastnosti napíše symbol *#*. K takto definované vlastnosti lze přistoupit pouze uvnitř dané třídy. V neposlední řadě může být také použita dědičnost k rozšíření vlastností jiné třídy. K tomu slouží klíčové slovo *extends*. Potomek, který je takto vytvořen, zdědí všechny vlastnosti i metody předka, včetně konstruktoru. [17] [18]

```
class MainTopic {
  constructor(idImage, idText) {
    //tridni promenne
    this.idImage = idImage;
    this.idText = idText;
    this.leftChildren = [];
    this.rightChildren = [];
    this.saveInfo = null;

    //tridni metody
    this.containsid = function(id) {
      if (id === this.idImage || id === this.idText) {
        return true;
      } else {
        return false;
      }
    };
```

Obrázek 3.1: Příklad použití třídy v JavaScriptu

Pro plné pochopení principu asynchronního programování je vhodné vysvětlit programování synchronní. Při zpracovávání synchronního kódu je program zpracováván jeden řádek za druhým tak, jak ho napsal programátor, přičemž kód je vykonáván tak rychle, jak je to jen možné. To znamená, že může proběhnout pouze jedna operace v jeden čas. JavaScript zpracovává pouze jedno vlákno, takže při zpracovávání jedné operace jsou všechny ostatní činnosti pozastaveny.

Asynchronní programování vnáší do zpracovávání kódu jistou míru paralelismu. Při zavolání asynchronní funkce nejsou výsledky k dispozici ihned a pro zachování plynulosti nemá smysl na tyto výsledky čekat. Místo toho se kód bude dále zpracovávat a je plně v režii programátora, aby zajistil, co se bude dít, až budou výsledky asynchronní funkce k dispozici.

V JavaScriptu existují dva hlavní typy asynchronního programování, callback a promise. Callback reprezentuje funkci předanou jako argument jiné funkci. Tato technika umožňuje funkci volat jinou funkci. Callback funkce může být použita i v rámci synchronního programování díky tomu, že jsou funkce objekty a lze je předat jako parametr funkce. Většinou ale bývá použita ke zpracování výsledků po proběhnutí nějaké asynchronní akce. Callback funkce bývá specifikována v argumentu asynchronní funkce, po jejímž provedení dojde k zavolání callback funkce, ve které programátor určí, co se stane po provedení asynchronní funkce.

```
function loadFromFile() {
  //height and width is in percent
  Office.context.ui.displayDialogAsync(
    "https://localhost:3000/src/dialog/fileinputpage.html",
    {
      height: 27,
      width: 27,
      displayInIframe: true
    },
    function(asyncResult) {
      dialog = asyncResult.value;
      dialog.addEventHandler(Office.EventType.DialogMessageReceived, processMessage);
    }
  );
}
```

Obrázek 3.2: Příklad použití callback funkce

Promise přístup představuje další novinku uvedenou v ECMA Script 2015 (ES6). Místo toho, abychom v argumentu volání funkce uvedli callback funkci, která bude zavolána po jejím vykonání, nám asynchronní funkce vrátí promise. Promise je objekt, který obsahuje informace o výsledku asynchronní operace. Tímto chováním nám prohlížeč v podstatě slibuje, že se nám vrátí výsledky dané funkce. Na tomto objektu můžeme zavolat funkci *then()*, která již obsahuje callback funkci a provede se ihned po tom, co se dokončí asynchronní operace. Tato funkce také obsahuje informaci o výsledku asynchronní operace. [19]

```
await OneNote.run(async function (context) {
  var page = context.application.getActivePage();

  //create image and text
  var topic = addMainTopic(page);
  var imgoutline = topic[0];
  var textoutline = topic[1];

  //load id
  imgoutline.load("id,paragraphs");
  textoutline.load("id");

  return context.sync().then(function () {
    topics = new MainTopic(imgoutline.id, textoutline.id);
    levelSettings.push(new Settings());
    initArrays();
    createDescription(context);
  });
}).catch(function (error) {
  handleError(error);
});
```

Obrázek 3.3: Příklad použití promise přístupu

S asynchronním programováním souvisí také dvě klíčová slova *async* a *await*. Klíčové slovo *async* se používá na začátku deklarace funkce a říká, že funkce by místo návratové hodnoty měla vracet promise. Klíčové slovo *await* může být použito pouze uvnitř funkce deklarované slovem *async*. Lze jej použít, pokud voláme funkci, která vrací promise. To způsobí, že se vykonávání kódu pozastaví, dokud nedojde k naplnění promise objektu. [20]

```
async function executeCreate(event) {
  await createMindMap();
  event.completed();
}
```

Obrázek 3.4: Příklad použití klíčových slov async a await

3.2 Hypertext Markup Language

Hypertext Markup Language (HTML), jak již název napovídá, je hypertextový značkovací jazyk používaný k vytváření obsahu webových stránek. HTML je spravováno a vyvíjeno organizací World Wide Web Consortium (W3C). Definuje strukturu a význam jednotlivých prvků webové stránky. Slovo hypertext v názvu znamená, že dokument obsahuje odkazy, které umožňují přesměrování na jiná místa ve stejném dokumentu, nebo dokonce na jiné dokumenty. Většina moderních webových stránek využívá HTML jako prostředek pro sdělování informací.

HTML používá takzvané značky k popisu toho, co se zobrazí na webovou stránku. Tyto značky zahrnují speciální elementy, které jsou složeny ze jména daného elementu ohraničeného ostrými závorkami. Příkladem elementu může být například element *<title>*, *<p>* nebo * *. Elementy můžou zahrnovat atributy, které obsahují další informace o tom, jak se má daný element zobrazit, jako například šířka a výška obrázku. [21] [22]

```
<!-- container pro výběr barvy textu -->
<div class="container">
  <label id="textColorLabel" for="textColor"></label>
  <br/>
  <input type="color" id="textColor" value="#000000" />
</div>
```

Obrázek 3.5: Příklad použití HTML elementů

3.3 Cascading Style Sheet

Cascading Style Sheet (CSS) reprezentuje jazyk pro popis vzhledu HTML dokumentu. CSS popisuje, jak by HTML elementy měly být zobrazeny. Pro tento účel lze využít i již dříve zmíněných atributů zahrnutých v HTML elementech, avšak v rámci osvědčených postupů je lepší se tomu vyhnout. Velkou výhodou stylování HTML dokumentu pomocí CSS je oddělení obsahu a struktury dokumentu od jeho grafické podoby. CSS lze ale také využít pro vytvoření pokročilejších animací či nejrůznějších přechodů a 3D transformací.

Vazbu mezi HTML elementem a stylem můžeme vytvořit na základě několika vlastností, například pomocí jména elementu, či jeho identifikátoru nebo třídy. Pravidlo pro popis vzhledu se tedy skládá ze selektoru, který identifikuje daný element, vlastnosti, kterou chceme nastavit a její hodnoty. [23]

```
.tabButton {
  color: #0078d7;
  border: none;
  outline: none;
  cursor: pointer;
  background-color: white;
  font-size: medium;
  margin: 5px;
  padding-bottom: 5px;
  margin-bottom: 5px;
}
```

Obrázek 3.6: Příklad používání css pravidel

3.4 Použité knihovny a API

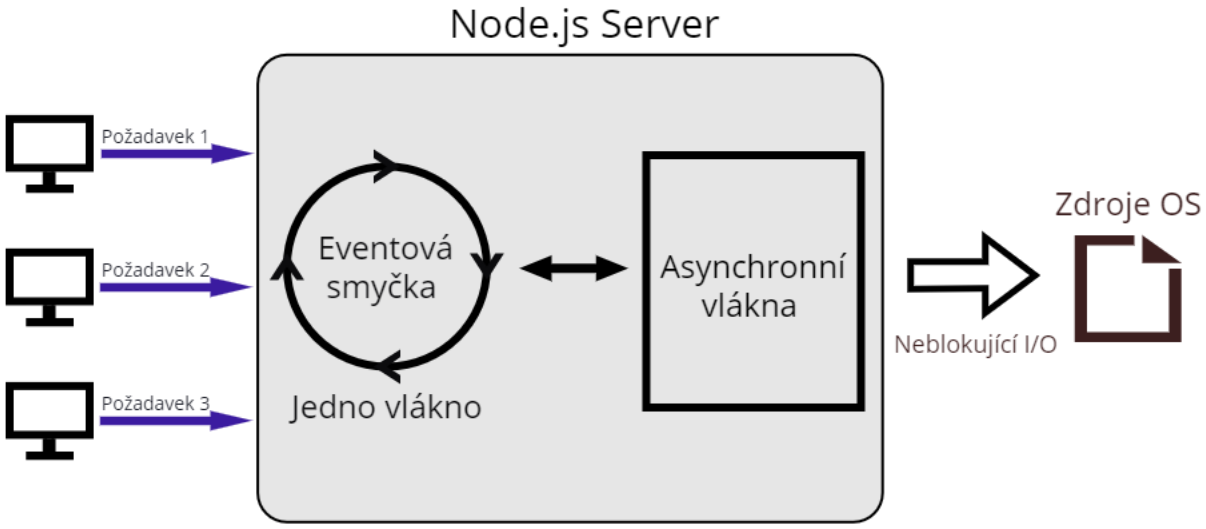
Knihovna je v programování označení pro soubor funkcí a procedur, jež je možno opakovaně použít více aplikacemi. S knihovnou úzce souvisí pojem API. Knihovny obvykle řeší nějaký problém, který se zprvu zdá být složitý a zabalí ho do svého kódu, který zprostředkovávají programátorovi pomocí aplikačního rozhraní (API). Ten tedy nemusí řešit problém, který za něj již vyřešil jiný programátor a může použít knihovnu, zatímco se soustředí na programování zbytku aplikace.

jQuery představuje velmi populární JavaScriptovou knihovnu, která usnadňuje práci při psaní kódu. Některé běžné postupy v JavaScriptu vyžadují mnoho řádků kódu. jQuery tyto postupy zabalí do jednoduchých metod, které se poté volají pomocí jediného řádku v kódu. Tím ve výsledku dochází ke zlepšení čitelnosti a zmenšení robustnosti zdrojového kódu. [24] [25]

```
$.getJSON(path, function(data) {
  createFromJSON(data);
});
```

Obrázek 3.7 Příklad použití knihovny jQuery

Node.js je běhové prostředí umožňující spouštět JavaScript mimo webový prohlížeč. Nejčastěji se používá k tvorbě serverové části webových aplikací. V důsledku toho Node.js umožňuje pohlížet na JavaScript nejen na jazyk, který se vykonává na straně klienta, ale i na jazyk, který je schopný běžet na straně serveru. Podobně jako JavaScript, má i Node.js architekturu řízenou událostmi využívající asynchronní operace. Je navržen tak, aby byl schopen obsluhovat co největší množství uživatelských požadavků. [26]



Obrázek 3.8: Architektura Node.js

Node Package Manager (npm), jak už název napovídá, je správce balíčků pro jazyk JavaScript. Původně byl vytvořen jako správce balíčků pro Node.js. Jeho jméno v názvu ovšem nese dodnes. Pomocí npm můžeme instalovat obrovské množství volně dostupných balíčků. Balíček je složen ze souborů, které definují nějakou funkcionalitu. Tyto soubory jsou označovány jako moduly. Modul je v JavaScriptu v podstatě knihovna, kterou můžeme zahrnout do jakéhokoliv projektu. Ke každému projektu či modulu se váže soubor package.json. Ten obsahuje základní informace o projektu, jako například verzi Node.js, na které daný projekt poběží nebo seznam balíčků, na kterých je daný projekt závislý.

Npm se ovládá z příkazové řádky, ku příkladu příkazem *npm install <package>* nainstalujeme balíček, který poté budeme využívat v kódu. [27] [28]

JSON reprezentuje jazykově nezávislý způsob zápisu dat, který slouží k uložení nějaké datové struktury ve formě textového řetězce. Vstupem může být číslo, řetězec, boolean, objekt nebo pole obsahující tyto hodnoty rozšířené o hodnotu null. JavaScript obsahuje zabudované funkce pro serializaci či deserializaci dat do nebo z JSON formátu. Pro převod dat do JSONu stačí zavolat funkci *JSON.parse()*, kdy v parametru uvedeme data, která chceme převést a jako výsledek této operace dostaneme již požadovaný textový řetězec. Obdobně funguje i převod z textového formátu na data, jen se o to stará funkce *JSON.stringify()*. [29]

Obrázek 3.9: Příklad JSON souboru

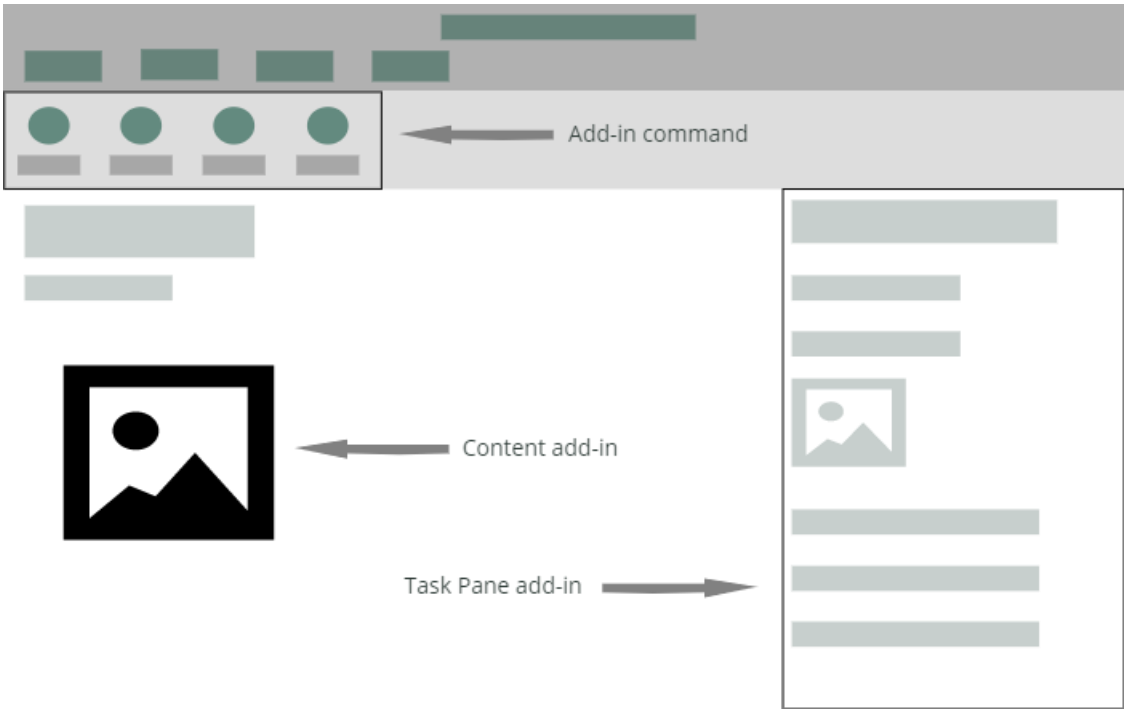
Jelikož je JSON textový formát, nelze do něj uložit objekty obsahující cykly. Objekt, který je použit pro uložení dat myšlenkové mapy ve formátu JSON, ale cykly obsahuje. K řešení tohoto problému lze použít knihovnu CircularJSON. Ta funguje velmi podobně jako funkce zabudované v JavaScriptu s tím rozdílem, že si umí poradit se zacyklenými objekty. Při převodu dat na textový řetězec knihovna hledá cykly. Pokud nějaký najde, označí ho znakem ~ a nahradí hodnotu, která uzavírá cyklus cestou od kořenového prvku k dané hodnotě. Použití této knihovny je prakticky stejné jako při použití normálního JSONu, pouze se místo JSON API volá CircularJSON API. [30]

3.5 Platforma Office doplňků

Office doplňky umožňují rozšířit funkci Office aplikací o nové možnosti a nový obsah. Pro jejich vytvoření je možno použít webové technologie HTML, CSS a JavaScript. Na rozdíl od doplňků vytvořených pomocí nástrojů VBA, COM a VSTO, což jsou starší řešení, která běží pouze v Office v systému Windows, Office doplňky nezahrnují kód, který běží na zařízení uživatele nebo v Office klientovi. Největší výhodou takového přístupu je jeho multiplatformní používání. Office doplňky lze spustit v systému Windows i Mac, v internetových prohlížečích, a dokonce i na iPadu.

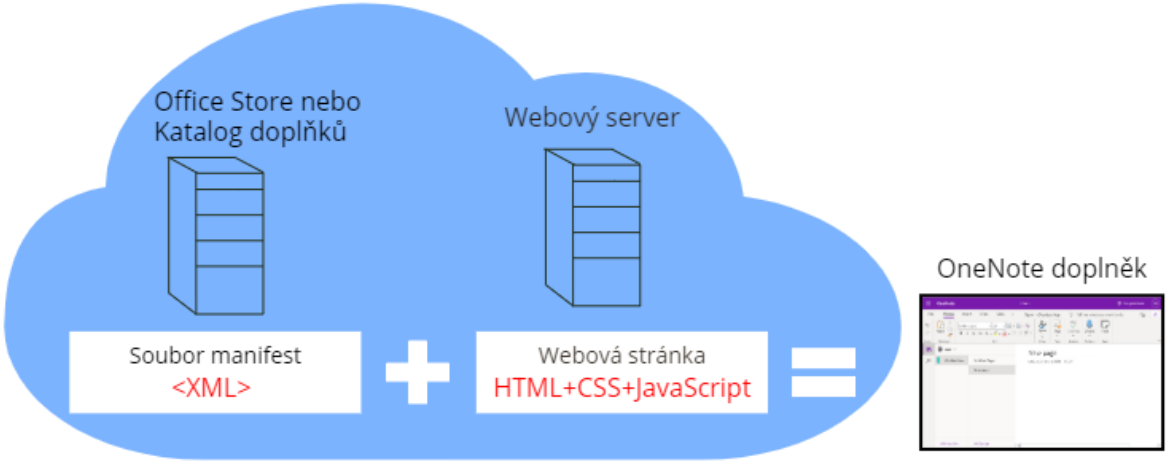
Moderní způsob vytváření integrovaných nástrojů pro aplikaci OneNote prostřednictvím jazyků HTML, CSS a JavaScript poskytuje tři způsoby, jak rozšířit OneNote. První možností je přizpůsobení pásu karet, který můžeme modifikovat přidáním nových tlačítek či menu. Tyto nově přidané prvky lze zobrazit buď v již existující kartě, nebo v nově vytvořené záložce. Dalším způsobem rozšíření OneNote aplikace je vložení interaktivního webového objektu uvnitř OneNote stránky. Takovým objektem může být například webová stránka, která poskytuje rozšířené možnosti interakce s OneNote aplikací. Posledním typem doplňku je zobrazení podokna na pravé části obrazovky, které

poskytuje téměř identické možnosti jako plnohodnotná webová stránka. Lze jej využít k zobrazení předem vytvořené webové stránky a její interakci s aplikací OneNote. [31]



Obrázek 3.10: Typy Office doplňků

Office doplňky obsahují dvě základní komponenty, soubor manifest typu XML a webovou aplikaci. Manifest soubor je uložen na straně klienta a specifikuje mimo jiné i URL webové stránky, kterou doplněk zobrazuje pomocí podokna na pravé části obrazovky. Webová aplikace se skládá z již dříve zmíněných technologií HTML, CSS a JavaScript. Tyto soubory jsou uloženy na webovém serveru a v aplikaci OneNote na webu se zobrazují prostřednictvím elementu *iframe*. Webová stránka je použita k hlubší interakci s uživatelem, než zahrnuje pouhé vytvoření tlačítka či menu v záložkách aplikace. Je možno ji zobrazit jako podokno, které poskytuje větší množství funkcí a lze k němu přistupovat jako ke klasické webové stránce. [32]



Obrázek 3.11: Kompozice Office doplňku

Soubor manifest.xml popisuje, jak by doplněk měl být použit po tom, co jej uživatel nainstaluje. Je složen z elementů, které nesou nějakou informaci a musí být v určitém pořadí, aby byla zajištěna platnost souboru. Obsahuje také různé řídící informace o doplňku, jako například jeho jméno, popis, verzi, výchozí jazykové nastavení, obrázky, které používá, jeho oprávnění, ale i to, jaká tlačítka nebo menu by měla být v aplikaci vytvořena, společně se všemi texty s nimi spojenými.

```
<Control xsi:type="Button" id="CreateButton">
  <!-- Text pro tlačítko -->
  <Label resid="CreateButton.Label"/>
  <!-- Definuje popisek daného prvku -->
  <Supertip>
    <Title resid="CreateButton.TitleTooltip"/>
    <Description resid="CreateButton.Tooltip"/>
  </Supertip>
  <!-- Ikona tlačítka. Nutno zadat ve třech různých velikostech -->
  <Icon>
    <bt:Image size="16" resid="IconCreate16"/>
    <bt:Image size="32" resid="IconCreate32"/>
    <bt:Image size="80" resid="IconCreate80"/>
  </Icon>
  <!-- Definuje akci po kliknutí na tlačítko -->
  <Action xsi:type="ExecuteFunction">
    <FunctionName>create</FunctionName>
  </Action>
</Control>
```

Obrázek 3.12: Příklad vytvoření tlačítka v souboru manifest.xml

Jak již bylo řečeno, Office doplněk se skládá z řídícího Manifest souboru a webové aplikace. To znamená, že veškerá funkčnost doplňku bude řešena ve zdrojovém kódu webové aplikace, tedy v kódu JavaScriptu. Ten může s OneNote stránkou komunikovat pomocí dvou různých API – OneNote JavaScript API a Common API. OneNote JavaScript API pokrývá většinu základní funkcionality, kterou budeme pro doplněk potřebovat. [33]

3.5.1 OneNote JavaScript API

OneNote JavaScript API představuje aplikačně-specifický objektový model pro OneNote, který byl představen společně s Office 2016. Pro klienty Office 2013 a starší není možno toto API použít. Jelikož doplněk reprezentuje webovou stránku uvnitř kontejneru prohlížeče v aplikaci Office, z důvodu výkonu není možné, aby doplněk s Office aplikací komunikoval synchronně. Místo toho je komunikace založena na Promise přístupu, který byl popsán dříve.

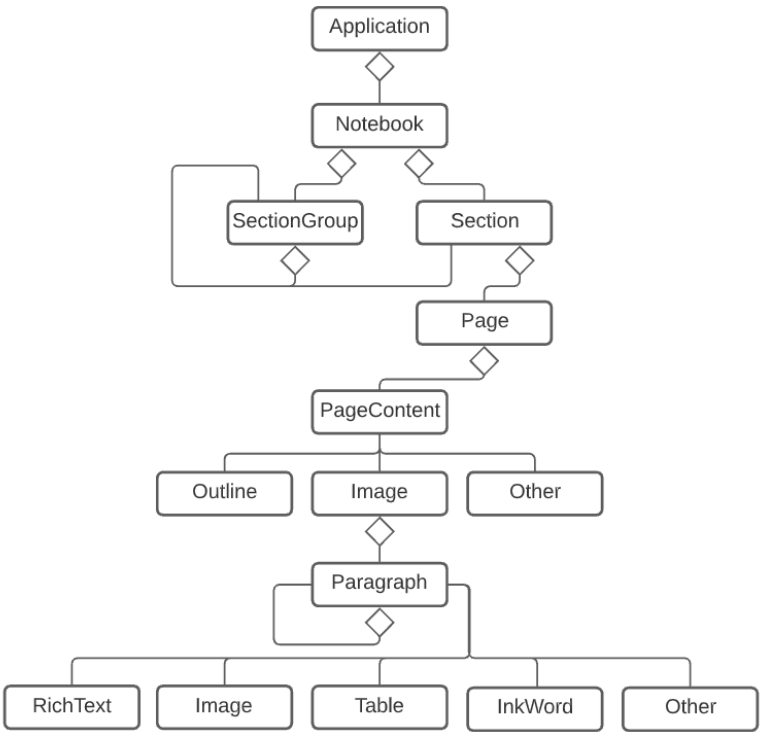
Protože OneNote aplikace a samotný doplněk fungují v jiných běhových prostředích, je nutno vytvořit objekt, jež zprostředkuje vzájemnou komunikaci. K tomu slouží funkce *OneNote.run()*. Ta automaticky vytvoří objekt *context*, který budeme používat k interakci s OneNote stránkou. Všechny objekty vytvořené uvnitř této funkce v rámci API jsou takzvané proxy objekty, přes které můžeme měnit vlastnosti OneNote objektů nebo volat jejich funkce. Jakákoliv akce, kterou provedeme na proxy objektu, je přidána do fronty. Po zavolání funkce *sync()* na objektu context je tato fronta odeslána do Office aplikace a provedena. Tato funkce se provádí asynchronně a vrací objekt Promise, který je naplněn po jejím provedení.

Pokud chceme zavolat nějakou funkci na proxy objektu, jednoduše ji zavoláme. Pro nastavení vlastnosti je nutno na proxy objektu, na kterém ji chceme nastavit, zavolat metodu *set*, přičemž a jejím argumentu musíme uvést buď další objekt stejného typu, který nastavujeme, nebo objekt s vlastnostmi, které chceme nastavit. Pokud chceme přečíst vlastnost OneNote objektu, musíme ji nejdříve explicitně načíst, aby se proxy objekt naplnil těmito daty. To provedeme zavoláním metody *load()* na daném objektu, přičemž v parametru uvedeme, kterou vlastnost chceme načíst. Poté před jejím přečtením musíme zavolat metodu *sync()*, aby došlo k naplnění Proxy objektu požadovanými daty. Po jejím vykonání již proxy objekt obsahuje požadovanou vlastnost a můžeme ji přečíst. [34]

```
await OneNote.run(function (context) {
  var outline = context.application.getActiveOutlineOrNull();
  outline.load("id");

  return context.sync().then(function () {
    var myTopic = getTopic(outline.id);
    if (myTopic == null) {
      console.log("Selected item is not image or text in Mind Map");
      return;
    }

    selectionChanged(myTopic);

    return context.sync();
  });
}).catch(function (error) {
  handleError(error);
});
```

Obrázek 3.13: Příklad používání OneNote JavaScript API

Pro práci obsahem stránky je používán objekt *Page*, který reprezentuje OneNote stránku. Na obrázku 3.14 je znázorněna struktura objektu, který představuje celou OneNote aplikaci.



Obrázek 3.14: Diagram objektového modelu OneNote

Objekt Page obsahuje id stránky, její název, ale hlavně kolekci objektů typu PageContent. Každý takový objekt má své id, pozici vůči osám x a y, a také objekty *Outline*, I*mage* a *Ink*. Dále obsahuje i vlastnost *type*, které specifikuje, který ze tří objektů je naplněn relevantními daty. Podle ní získáme objekt daného typu. Doplněk pracuje pouze s objekty typu *Outline*, které jsou složeny z odstavců či obrázků. Ty již obsahují objekty, které vidíme ve OneNote stránce. V našem případě jsou jimi objekty *Image* nebo *RichText*.

Do stránky lze přidávat pouze výčet HTML elementů, které poté OneNote stránka zpracuje a vytvoří vlastní obsah. Mezi ně patří například elementy *<p>*, *<img>* nebo * *. Každý z těchto elementů může mít nastavené některé atributy, jež se projeví změnou jeho vzhledu. U elementu <img> je to například vlastnost *width*, *height* nebo *src*. K přidávání objektů do stránky slouží metody *Page.addOutline* a *Outline.appendHtml*. Metoda addOutline obsahuje tři parametry. Prvníma dvěma jsou souřadnice x a y, kam bude daný prvek vložen. Třetí parametr je HTML řetězec, který popisuje, co za prvek se vloží na stránku. Metoda appendHtml má pouze jeden parametr, který říká, co za HTML element se vloží na konec již existujícího prvku ve stránce. [35]

```
async function addSubtopicElements(page, context, myTopic, direction, shift, depth) {
  //create line html and its coordinates
  var line = await calcLineCoords(context, myTopic, direction, shift);

  //calc x and y coordinates where will be inserted image and text and creates their html
  var obj = calcImageAndTextCoords(line.x, line.y, direction, shift, depth);

  //add html elements to OneNote page
  var imgoutline = page.addOutline(obj.imageX + 1, obj.imageY + 1, obj.imageHtml);
  var textoutline = page.addOutline(obj.textX, obj.textY, obj.textHtml);
  var lineoutline = page.addOutline(line.x, line.y, line.html);

  return [imgoutline, textoutline, lineoutline];
}
```

Obrázek 3.15: Příklad přidání nového tématu na OneNote stránku

3.5.2 Common API

Common API představuje aplikační rozhraní, které je možno používat napříč téměř všemi Office aplikacemi. Je podporováno od Office 2013. Jelikož bylo vytvořeno dřív, než byl představen ES6, který zavedl Promise přístup, funguje toto API na principu callbacků. Zavádí rozšířené možnosti interakce s Office aplikací, jako jsou dialogová okna, nastavení uložena v dokumentu nebo manipulaci s aktivním objektem ve stránce. Common API je navrženo tak, aby doplňovalo aplikačně specifické API. To znamená, že by se mělo používat pouze v případě, že pro danou funkcionalitu není možno aplikačně specifické API použít. [36]

Podobně, jako je to u OneNote JavaScript API, i Common API má k dispozici objekt *context*, který je dostupný ihned po inicializaci doplňku. Ten se sice liší od context objektu, který je k dispozici pro OneNote JavaScript API, ale princip jeho použití a významu je velmi podobný. Mimo to jsou zde i jiné důležité objekty, jako například *Document* nebo *Mailbox*. Objekt *Document* slouží k práci se souborem, který představuje daný dokument, čtení či zápis dat do aktuálně zvoleného prvku nebo pro přístup k objektu Settings. Velké množství těchto funkcí jsou ale podporováno pouze některými Office aplikacemi. [37]

OneNote toto API podporuje jen velmi okrajově. Common API je ve OneNote možno použít pouze k zaregistrování posluchače na událost *DocumentSelectionChanged*, která nastane pokaždé, když se na stránce změní zvolený objekt. Dále Common API slouží k zavolání funkcí *getSelectedDataAsync*, *setSelectedDataAsync*, případně pro přístup k objektu *Settings*. První dvě funkce lze použít k již zmíněnému získání či nastavení dat, která jsou obsažena v právě aktivním prvku. Objekt *Settings* se stará o uložení nastavení v rámci dokumentu. To lze použít k uložení primitivních hodnot ve formátu klíč hodnota. Je možné jej ale použít pouze pro doplňky typu Content add-in. [38]

3.5.3 Možnosti nasazení OneNote doplňků

Způsobů, jak publikovat doplněk mezi koncové uživatele je hned několik. Každý z nich má ovšem jinou cílovou skupinu uživatelů. Některé způsoby je vhodné použít, pokud chceme doplněk zveřejnit pouze v rámci nějaké naší organizace, jiné zase, pokud chceme doplněk zpřístupnit prostřednictvím Office Store, kde si jej může stáhnout jakýkoliv uživatel. My si shrneme způsob, jak publikovat doplněk na Office Store. [39]

Pro úspěšné zveřejnění doplňku je nutné, aby stránka, která je v doplňku používána, byla dostupná z internetu, stejně tak jako všechny další zdroje, které používá. K zahájení celého procesu musí vývojář nejprve odeslat aplikaci do partnerského centra Microsoftu, přičemž musí být přihlášen pod vývojářským účtem. Zde musí vyplnit informace jako název doplňku, jeho jazykové nastavení, datum, od kdy bude doplněk dostupný, podmínky použití, zásady ochrany osobních údajů a v neposlední řadě také manifest soubor. Jakmile vývojář uvede všechny požadované informace a potvrdí je, publikování doplňku se bude nacházet ve fázi schvalování. Poté, co je tento proces dokončen, může si vývojář zobrazit jeho výsledek. Pokud byl proces úspěšný, doplněk se v dříve specifikovaném čase objeví v obchodě pro Microsoft doplňky. [40] [41]

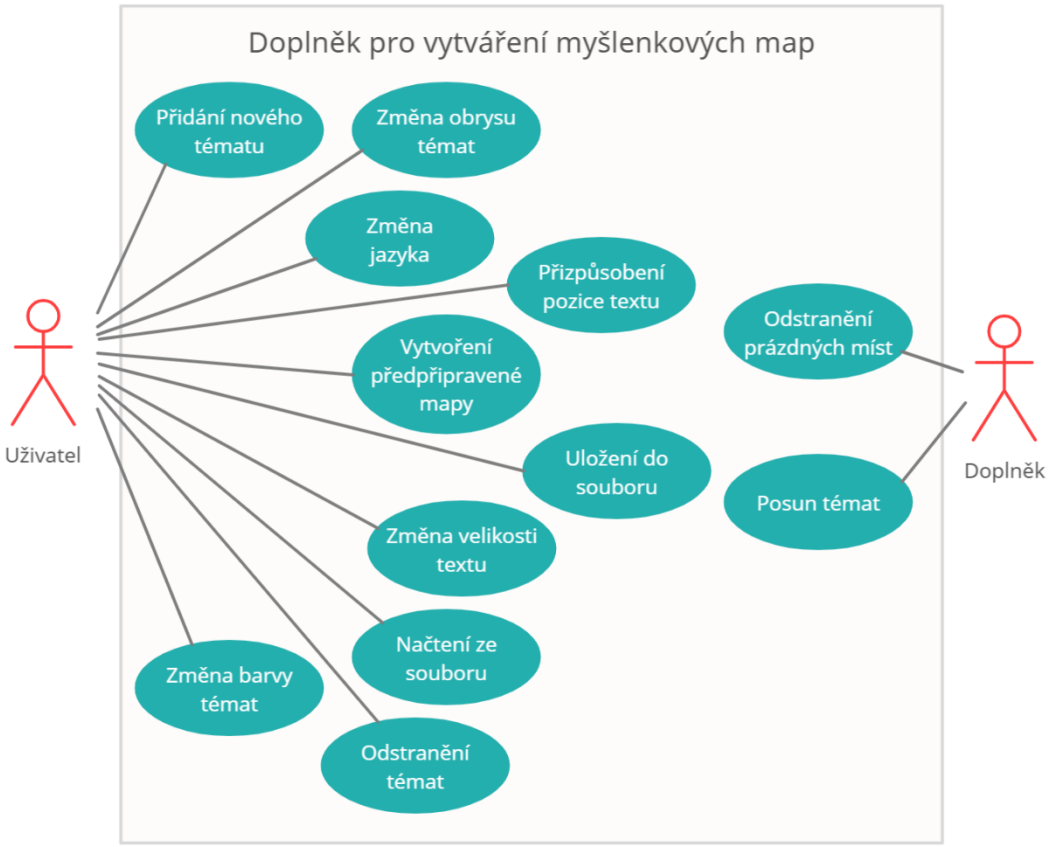
Kapitola 4

Vytvoření vlastního doplňku

V této části rozebereme jednotlivé funkce, které doplněk podporuje a bude zde popsána architektura, která byla zvolena pro implementaci doplňku. Na závěr kapitoly budou uvedeny možnosti, jak doplněk rozšířit. V předchozí kapitole byly částečně vysvětleny technologie a principy, které se používají pro vytváření myšlenkové mapy. Tato kapitola na ně bude navazovat a rozvíjet jejich popis.

4.1 Specifikace funkcí doplňku a jeho uživatelské rozhraní

Při navrhování funkcí doplňku byl brán zřetel na to, aby bylo vytváření myšlenkové mapy snadné a přirozené a aby doplněk co nejvíce usnadňoval práci uživateli. Níže je vyobrazen use case diagram, který definuje všechny funkce, které doplněk poskytuje společně s tím, kdo je za danou funkci zodpovědný. [48]



Obrázek 4.1: Use case diagram znázorňující funkce doplňku

Přidání nového tématu reprezentuje hlavní funkci doplňku, bez které se neobejde žádný nástroj pro vytváření myšlenkových map. Slouží k vytvoření nového tématu, jež bude spojeno s uživatelem zvoleným tématem. Pokud není možno dané téma vytvořit kvůli tomu, že jej není kam vložit, dojde k vyvolání funkce posun témat, která bude popsána později. Níže je vyobrazen scénář případu užití popisující jednotlivé kroky, jež se provádějí v rámci daného případu užití.

Tabulka 4.1: Scénář případu užití přidání nového tématu

Název případu užití	Přidání nového tématu
Popis	Případ užití popisuje přidání nového podtématu k uživatelem zvolenému tématu.
Aktéři	Uživatel
Podmínky pro spuštění	Uživatel vytvořil myšlenkovou mapu a chce ji upravit.
Základní tok: 1. Uživatel zvolí téma, které chce rozšířit. 2. Uživatel klikne na ikonu „Přidat téma“. 3. Doplněk z popisku hlavního tématu načte objektovou strukturu myšlenkové mapy. 4. Doplněk získá souřadnice tématu, na které uživatel kliknul a vypočte souřadnice pro přidání nového tématu. 5. Doplněk vytvoří HTML řetězce reprezentující obrys, text a šipku nového tématu. 6. Doplněk přidá na vypočtené souřadnice všechny objekty, které reprezentují nové téma. 7. Doplněk aktualizuje strukturu myšlenkové mapy. 8. Doplněk zapíše novou objektovou strukturu do popisku hlavního tématu.	
Alternativní tok 1: 3.1 Doplněk nenalezne objekt obsahující popisek, který představuje strukturu myšlenkové mapy. 3.2 Doplněk ukončí přidávání nového tématu.	
Alternativní tok 2: 3.1 Doplněk zjistí, že objekt, na který uživatel kliknul není součástí myšlenkové mapy. 4.2 Tímto případ užití končí.	
Stav myšlenkové mapy po ukončení případu užití	Bylo přidáno nové téma do myšlenkové mapy a byla aktualizována její struktura.

Přizpůsobení textu je velmi užitečná funkce, která slouží k vycentrování veškerého textu tak, aby byl uprostřed daného obrysu tématu. Pokud tedy uživatel mění výchozí text jednotlivých témat a nechce se mu složitě pohybovat s každým textem tak, aby byl uprostřed tématu, jednoduše může kliknout na tlačítko, které provede tuto funkci a všechny texty se samy vycentrují. Princip jejího provádění spočívá v tom, že se nejdříve musí načíst a dočasně uložit všechny souřadnice i rozměry obrysů témat a jejich texty. Poté se na základě velikosti fontů, délky textů a velikosti i rozměrů obrysu vypočtou nové souřadnice všech textů. Nyní už stačí pouze změnit pozice x a y těchto textů na předem vypočtené hodnoty.

Odstranění témat představuje další základní funkci, bez které se neobejde žádná myšlenková mapa. Tu je možno vyvolat bud tlačítkem „Delete Topic“ nebo tlačítkem „Delete All“. Jediný rozdíl mezi

těmito tlačítky je ten, že tlačítko „Delete Topic“ smaže právě zvolené téma a s tím i všechna jeho podtémata. Naopak tlačítko „Delete All“, jak již název napovídá, smaže celou myšlenkovou mapu bez ohledu na to, které téma je právě zvoleno. Pro úspěšné odstranění témat je nejdříve nutno získat téma, které chce uživatel smazat. Pokud byla tato funkce zavolána po kliknutí na tlačítko „Delete All“, je tímto tématem hlavní téma. Jestliže funkce byla zavolána po kliknutí na tlačítko „Delete Topic“, je nutno nejdříve získat id aktuálně zvoleného tématu a poté s jeho pomocí najít objekt daného tématu. Dále je třeba smazat téma ze tří různých míst. Z OneNote stránky, z globálního pole udávajícího obsazenost daného místa na stránce a ze stromové struktury, jež obsahuje všechny důležité informace o tématu. Poté, co bylo téma společně se svými potomky kompletně odstraněno, spustí se funkce, která zjistí, jestli v myšlenkové mapě nevznikly nějaké prázdné řádky. Pokud ano, funkce tyto řádky smaže a posune témata tak, aby vyplnila tyto vzniklé mezery. Tato funkce je dále popsána scénářem případu užití, jež detailně popisuje průběh vykonávání této funkce.

Tabulka 4.2: Scénář případu užití odstranění tématu

Název případu užití	Odstranění tématu
Popis	Případ užití popisuje odstranění uživatelem zvoleného tématu a všech jeho podtémat.
Aktéři	Uživatel
Podmínky pro spuštění	Uživatel vytvořil myšlenkovou mapu a chce odstranit téma.
Základní tok: 1. Uživatel zvolí téma, které chce smazat. 2. Uživatel klikne na ikonu „Smazat téma“. 3. Doplněk z popisku hlavního tématu načte objektovou strukturu myšlenkové mapy. 4. Doplněk načte ID tématu, na které uživatel kliknul a s jeho pomocí vyhledá objekt, který reprezentuje dané téma ve stromové struktuře. 5. Doplněk z OneNote stránky odstraní obrys, text i šipku směřující k danému tématu, společně se všemi jeho podtématy. 6. Doplněk aktualizuje globální pole, které uchovává obsazenost jednotlivých pozic myšlenkové mapy. 7. Doplněk smaže uživatelem vybrané téma z globální stromové struktury, reprezentující témata myšlenkové mapy. 8. Pokud se v myšlenkové mapě vyskytla prázdná místa, doplněk pozmění strukturu myšlenkové mapy tak, aby vzniklá mezera zanikla. 9. Doplněk zapíše novou objektovou strukturu do popisku hlavního tématu.	
Alternativní tok 1: 3.1 Doplněk nenalezne objekt obsahující popisek, který představuje strukturu myšlenkové mapy. 3.2 Případ užití je ukončen.	
Alternativní tok 2: 4.1 Doplněk zjistí, že objekt, na který uživatel kliknul není součástí myšlenkové mapy. 4.2 Tímto případ užití končí.	
Stav myšlenkové mapy po ukončení případu užití	Bylo odstraněno téma z myšlenkové mapy a byla aktualizována její struktura.

Velmi užitečnou funkcí, kterou ocení například učitelé, kteří chtějí svým žákům rozeslat myšlenkovou mapu, aby ji mohli všichni upravit podle sebe a následně ji odevzdat, je uložení, případně načtení myšlenkové mapy ze souboru. Pro uložení mapy do souboru je třeba načíst data myšlenkové mapy z databáze, získat text každého tématu a následně tyto proměnné uložit do jednoho JSON objektu, který si uživatel stáhne ve formě JSON souboru. K vytvoření myšlenkové mapy ze souboru stačí od uživatele získat již dříve vytvořený JSON soubor, ze kterého získáme všechny proměnné reprezentující myšlenkovou mapu a pomocí kterých ji můžeme celou vytvořit. Pro vytvoření myšlenkové mapy z proměnných je nejprve nutno vypočítat souřadnice x a y pro všechna témata a následně vytvořit html řetězce, které budou přidány do OneNote stránky. Poté již je třeba jen přidat tyto řetězce do stránky a načíst jejich id, aby následně mohla být zaměněna za stávající id, která již nejsou aktuální.

Vytvoření předpřipravené myšlenkové mapy je vhodné pro uživatele, kteří chtějí vytvořit jednoduchou myšlenkovou mapu co nejrychleji. Uživatel má na výběr ze tří předpřipravených myšlenkových map, a to malých, středních a velkých rozměrů. Po tom, co uživatel zvolí požadovanou velikost myšlenkové mapy, se načte jeden ze tří souborů, ve kterých jsou uložená data potřebná pro vytvoření celé myšlenkové mapy. Dále se postupuje stejně, jako při funkci načtení myšlenkové mapy ze souboru, která již byla popsána výše. Níže je popsán scénář tohoto případu užití.

Tabulka 4.3: Scénář případu užití vytvoření předpřipravené myšlenkové mapy

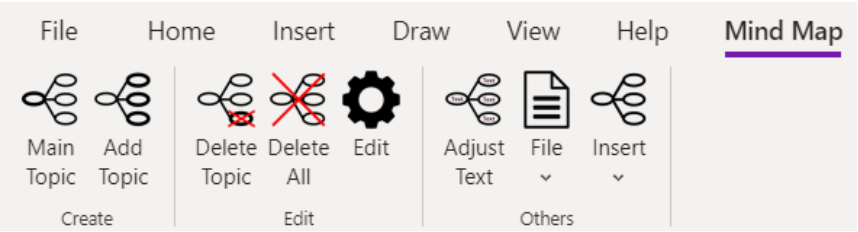
Název případu užití	Vytvoření předpřipravené myšlenkové mapy
Popis	Případ užití popisuje vytvoření vzorové myšlenkové mapy.
Aktéři	Uživatel
Podmínky pro spuštění	V rámci OneNote stránky neexistuje žádná myšlenková mapa.
Základní tok: 1. Uživatel klikne na ikonu „Vložit“, čímž doplněk nabídne možnost vybrat velikost vzorové myšlenkové mapy. 2. Uživatel zvolí požadovanou velikost myšlenkové mapy. 3. Doplněk načte ze serveru JSON soubor, který obsahuje definici myšlenkové mapy. 4. Doplněk přečte soubor a na základě v něm obsažených dat naplní stromovou strukturu a další proměnné reprezentující myšlenkovou mapu. 5. Doplněk pomocí těchto dat spočítá souřadnice, kam budou vložen objekty z myšlenkové mapy. 6. Doplněk vytvoří HTML řetězce reprezentující obrys, text a šipku nového tématu. 7. Doplněk přidá na vypočtené souřadnice všechny objekty myšlenkové mapy. 8. Doplněk načte ID všech vytvořených objektů. 9. Doplněk nahradí původní ID nově načtenými. 10. Doplněk zapíše novou objektovou strukturu do popisku hlavního tématu.	
Alternativní tok 1: 3.1 Ze serveru se nepodařilo načíst soubor. 3.2 Případ užití končí.	
Stav myšlenkové mapy po ukončení případu užití	Byla vytvořena vzorová myšlenková mapa.

Poslední funkcí, která je plně v režii uživatele, je editace vzhledu myšlenkové mapy. Doplněk, jak již bylo zmíněno, umožňuje editovat, barvu i pozadí textu a témat, typ obrysu i velikost textu. Jako typ obrysu lze zvolit buď elipsu, obdélník nebo kosočtverec. Při změně některé z těchto vlastností je nutno nejdříve najít témata, která budou změněna. Ta získáme na základě toho, jestli uživatel chce změnit vlastnost jednoho tématu nebo celé úrovně najednou. Poté musí dojít k načtení souřadnic x a y, rozměrů a případně i textu objektů, které budou změněny. Tyto objekty budou následně smazány a na jejich místo se vytvoří nové objekty s již nově definovaným vzhledem. Pokud uživatel změnil vzhled textu, bude opětovně vytvořen pouze text a obrázek zůstane tak, jak je. V případě změny barvy či typu obrysu musí dojít ke smazání a následnému vytvoření textu i obrysu, protože by jinak obrys mohl překrýt text, který by nemusel být vidět, pokud by obrys měl barevné pozadí. Posledním krokem je nahrazení starých id novými tak, jak již bylo zmíněno u předchozích funkcí.

Pokud se při vytváření témat dostane uživatel do situace, kdy přidání nového tématu blokuje jiné téma, je třeba takové téma posunout tak, aby se uvolnilo místo pro nové téma. To zajistí funkce, která se stará o posun témat. Kvůli zachování rovnoměrné struktury myšlenkové mapy bude vždy posouváno téma první úrovně, tedy téma, které je přímým potomkem hlavního tématu, společně se všemi jeho potomky. Poté, co najdeme téma první úrovně, které mezi svými potomky obsahuje téma, jenž blokuje vytvoření nového tématu, musíme zjistit, jestli můžeme posunout toto téma. Pokud ne, musíme najít další téma první úrovně, jehož potomek blokuje posunutí tohoto tématu, a celou akci zopakovat. Jakmile budeme mít seznam témat, která je třeba posunout bez toho, aby se nějaká témata překrývala, jednoduše načteme souřadnice x a y všech objektů, ze kterých se tato témata skládají a připočteme k nim vertikální vzdálenost mezi jednotlivými tématy.

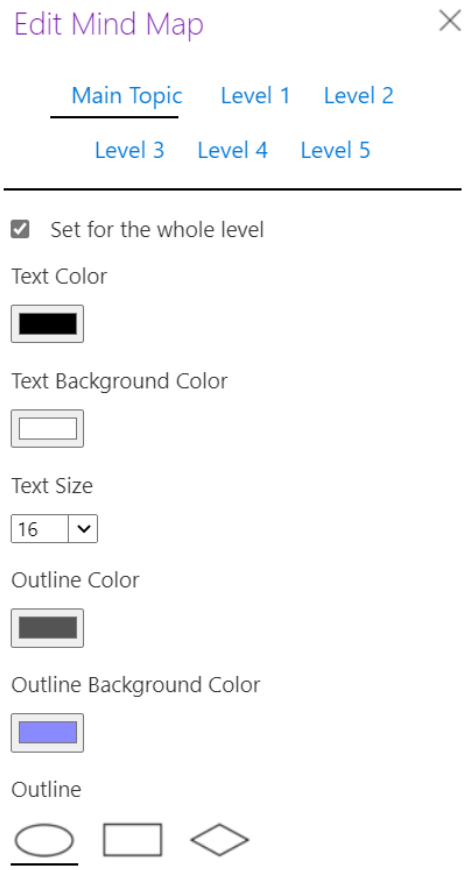
Dále doplněk podporuje zobrazení v českém i anglickém jazyce. Volba jazyka se provede na základě toho, v jakém jazyce je aplikace OneNote právě zobrazena. Kompletní změna jazyka zahrnuje změnu textů v pásu karet, dialogovém okně a v podokně, ve kterém se mění vzhled myšlenkové mapy.

Doplněk nabízí několik možností interakce s uživatelem, pomocí kterých lze vykonat jednotlivé funkce. Tou první je nově vytvořená záložka v pásu karet, která obsahuje menu a tlačítka, odkud lze provést většinu akcí spojených s vytvářením myšlenkové mapy.



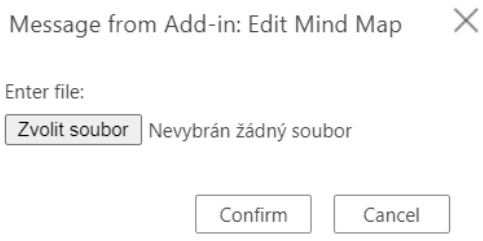
Obrázek 4.2: Menu a tlačítka dostupná ze záložky v pásu karet

Druhou možností, jak může uživatel upravovat myšlenkovou mapu, je podokno na pravé straně obrazovky, které lze otevřít kliknutím na tlačítko „Edit“ v záložce „Mind Map“. Slouží ke změně velikosti textu, barvy textu nebo témat či jejich pozadí nebo změně typu obrysu. Tyto atributy, jak již bylo zmíněno, lze nastavit jak pro celou úroveň najednou, tak pro jednotlivá témata.



Obrázek 4.3: Podokno pro editaci myšlenkové mapy

Doplněk nabízí ještě jedno rozhraní, jež lze využít pro komunikaci s uživatelem. Je jím dialogové okno umožňující nahrát soubor, který bude využit k vytvoření nové myšlenkové mapy. Toto okno je pro načtení souboru od uživatele nezbytné, jelikož kliknutím na tlačítko v záložce karet se pouze zavolá funkce, na jejímž základě ale nelze otevřít okno pro výběr souboru, protože se nejeví jako akce vyvolaná uživatelem.



Obrázek 4.4: Dialogové okno pro výběr souboru

4.2 Návrh a architektura doplňku

Při návrhu doplňku a jeho následném vývoji byly dodržovány standardní postupy softwarového inženýrství, které si kladou za cíl navrhnout kvalitní, udržovatelný a snadno rozšiřitelný software. Použité metody jsou popsány v následujících publikacích, viz [42] [43] [44].

V případě, že je doplněk spuštěný, jsou všechny zdroje, včetně webových stránek a obrázků, které doplněk využívá, uloženy na Node.js serveru. Aby bylo možné doplněk používat, musí být server zapnutý. Lokální webový server lze spustit za více účely, mezi které patří například vývoj doplňku nebo jeho testování. K jeho spuštění slouží již předpřipravené skripty, jež jsou součástí vzorového projektu a které lze spustit příkazem *npm run <script-name>*, přičemž jako jméno skriptu lze použít jednu z hodnot uvedenou v souboru package.json pod klíčem *scripts*. [45]

```
"scripts": {
  "build": "webpack -p --mode production --https false",
  "build:dev": "webpack --mode development --https false",
  "build-dev": "webpack --mode development --https false && echo . && echo .
  "dev-server": "webpack-dev-server --mode development",
  "lint": "office-addin-lint check",
  "lint:fix": "office-addin-lint fix",
  "prettier": "office-addin-lint prettier",
  "start": "office-addin-debugging start manifest.xml",
  "start:desktop": "office-addin-debugging start manifest.xml desktop",
  "start:web": "office-addin-debugging start manifest.xml web",
  "stop": "office-addin-debugging stop manifest.xml",
  "validate": "office-addin-manifest validate manifest.xml",
  "watch": "webpack --mode development --watch"
},
```

Obrázek 4.5: Ukázka všech skriptů, které lze spustit příkazem npm run v rámci jednoho projektu.

Témata v myšlenkové mapě jsou uspořádána do stromové hierarchie, kdy z jednoho nadřazeného tématu vychází vždy libovolný počet podtémat. Podobný námět byl použit pro vytvoření vnitřní struktury, která reprezentuje celou myšlenkovou mapu. Tu tvoří jediná globální proměnná, do které se po vytvoření myšlenkové mapy zapíše objekt typu *MainTopic*. Ten obsahuje pole témat, která vychází z hlavního tématu myšlenkové mapy. Tato témata jsou instancí třídy *Topic* a opět zahrnují pole témat stejného typu. Ve výsledku tato jediná globální proměnná typu *MainTopic* tvoří kořen stromové struktury, zatímco všechna ostatní témata jsou typu *Topic* a tvoří zbylé vrcholy daného stromu. Taková struktura je velmi podobná fyzickému rozložení témat v myšlenkové mapě.

Třída *MainTopic* je složena z následujících vlastností:

- idImage: id obrázku reprezentujícího obrys hlavního tématu
- idText: id objektu, který obsahuje text hlavního tématu
- leftChildren: pole témat vycházejících z hlavního tématu směrem na levou stranu
- rightChildren: pole témat vycházejících z hlavního tématu směrem na pravou stranu
- saveInfo: do této proměnné bude zapsán objekt obsahující informace o velikosti textu, barvách a typu obrysu daného tématu pouze v případě, že se liší od fontu a barev, které jsou společné pro všechna témata na dané úrovni

Třída *Topic* se sice na první pohled příliš neliší od třídy *MainTopic*, ve skutečnosti ale obsahuje mnohem více metod. Její struktura vypadá následovně:

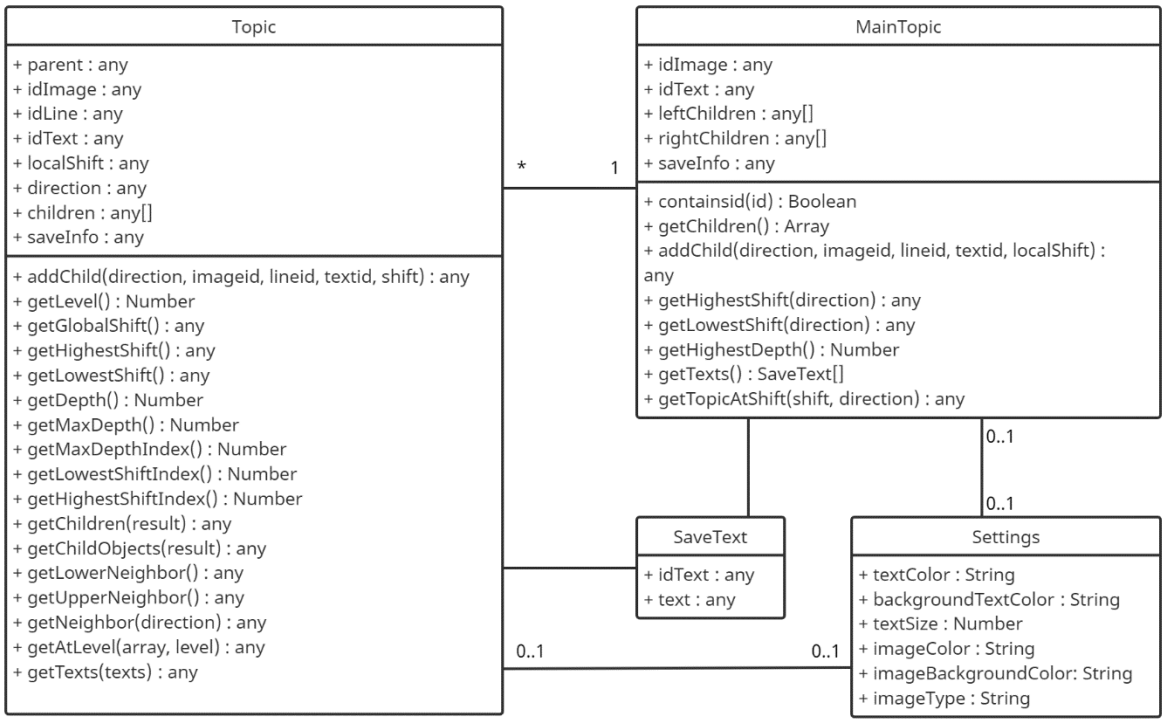
- parent: odkazuje na nadřazené téma
- idImage: id obrázku reprezentujícího obrys hlavního
- idText: id objektu, který obsahuje text hlavního tématu
- idLine: id obrázku představujícího čáru vedoucí od nadřazeného k tématu směrem k tomuto tématu
- localShift: znázorňuje vertikální posun vůči nadřazenému tématu
- direction: udává, jestli je téma na pravou či levou stranu od hlavního tématu
- children: pole témat, která vycházejí z tohoto tématu
- saveInfo: do této proměnné bude zapsán objekt obsahující informace o velikosti textu, barvách a typu obrysu daného tématu pouze v případě, že se liší od fontu a barev, které jsou společné pro všechna témata na dané úrovni

Dalším prvkem popisujícím existující myšlenkovou mapu je třída *Settings*. Ta slouží k uchování velikosti textu, typu obrysu a barvy popředí a pozadí, které mají texty nebo obrysy. Nastavení těchto vlastností je možno provést dvěma způsoby. Uživatel může změnit tyto atributy buď jedinému tématu, nebo všem tématům v dané úrovni. Každá myšlenková mapa je rozdělena do několika úrovní v závislosti na tom, jak moc jsou její témata zanořená. Všechna témata, která vedou z hlavního tématu, tvoří první úroveň. Každé téma vedoucí z témat první úrovně, je téma druhé úrovně, a tak dále. Instance třídy *Settings* jsou uloženy v globálním poli, kdy každý jeho prvek reprezentuje jednu úroveň v myšlenkové mapě. Pokud tedy uživatel změní vlastnost témat v dané úrovni, dojde k modifikaci tohoto globálního pole. Druhým způsobem, jak lze použít třídu *Settings*, je změna atributu pouze jednoho tématu. To způsobí zápis nové proměnné typu *Settings* do třídní proměnné *saveInfo*, kterou mají všechna témata. Ta má například při vytváření myšlenkové mapy ze souboru přednost před globální proměnnou, která určuje vlastnosti témat v celé úrovni.

Seznam vlastností, které obsahuje třída *Settings* a které lze změnit u každého tématu:

- textColor: barva textu
- backgroundTextColor: barva pozadí textu
- textSize: velikost fontu textu
- imageColor: barva obrysu tématu
- backgroundImageColor: barva výplně tématu
- imageType: typ obrysu tématu

Na následujícím obrázku je vyobrazen třídní diagram, který podrobně popisuje všechny třídy, jež jsou vytvořeny v rámci doplňku.



Obrázek 4.6: Třídní diagram zobrazující třídy používané v doplňku [46]

Poté, co se vytvoří nové téma, je potřeba zapsat, že na dané pozici již není volno, aby zde nebylo přidáno další téma. K tomuto účelu slouží dvě globální dvourozměrná pole, kdy každé slouží pro jednu stranu myšlenkové mapy. Na začátku se tato pole vytvoří o velikosti deset krát deset a inicializují se na hodnotu nula. V případě, že dojde k naplnění některého z polí, dojde k zdvojnásobení jejich velikosti tak, aby to nijak nenarušilo přidávání dalších témat. Pokud uživatel vytvoří téma, tak nejdříve dojde ke kontrole, jestli ho lze na požadovanou pozici vložit. Pokud ano, tak se na danou pozici v poli zapíše id tématu, které je rodičem právě přidaného tématu, čímž zaručíme, že zde nebude vloženo další téma. Pokud se na pozici, na kterou má být vloženo nové téma, nenachází nula, znamená to, že je na daném místě již vytvořeno nějaké téma a musí dojít k jeho posunutí tak, aby se vytvořilo místo pro nové téma. Tato funkcionalita bude popsána později. Poté, co se téma posune, a tím se vytvoří místo pro nově přidané téma, je možno jej na dané místo zapsat.

Tento postup, kdy je ověřováno, jestli lze na dané místo přidat téma, lze provádět bez použití právě popsaných globálních polí. Je možno ho nahradit procházením celé struktury témat a současným kontrolováním, jestli se na dané pozici nenachází nějaké téma. Tato metoda je ale v případě velkého zanoření témat příliš zdlouhavá a komplikovaná. Její výhodou je ale to, že nevyžaduje přenášet žádná explicitní data mezi OneNote aplikací a doplňkem.

Jak již bylo zmíněno v předchozí kapitole, OneNote aplikaci lze rozšířit třemi způsoby, přičemž nástroj pro vytváření myšlenkových map používá dva z nich. Jedná se o podokno v pravé části obrazovky a nově přidanou záložku v pásu karet, která slouží k otevření podokna, a navíc definuje vlastní tlačítka a menu. Po kliknutí na tlačítko v záložce karet dojde k vykonání funkce, jejíž název je specifikován v souboru manifest.xml. Pro tento účel existuje v rámci doplňku neviditelná webová stránka, na které bude tato funkce provedena. Ta ale nijak nekomunikuje se stránkou ve formě

podokna na pravé části obrazovky, která se navíc neinicializuje společně s vložením doplňku do aplikace či při načtení OneNote stránky. To znamená, že stránka, která zajišťuje veškerou funkcionalitu doplňku, existuje dvakrát. Pokud uživatel přidá pomocí tlačítka v záložce karet nové téma myšlenkové mapy, tato událost nastane pouze v rámci neviditelné stránky, která reprezentuje část doplňku představující záložku s tlačítky a menu. Podokno pro editaci témat tedy nijak nezjistí, že uživatel již vytvořil myšlenkovou mapu a měl by mít možnost editovat její vzhled. [47]

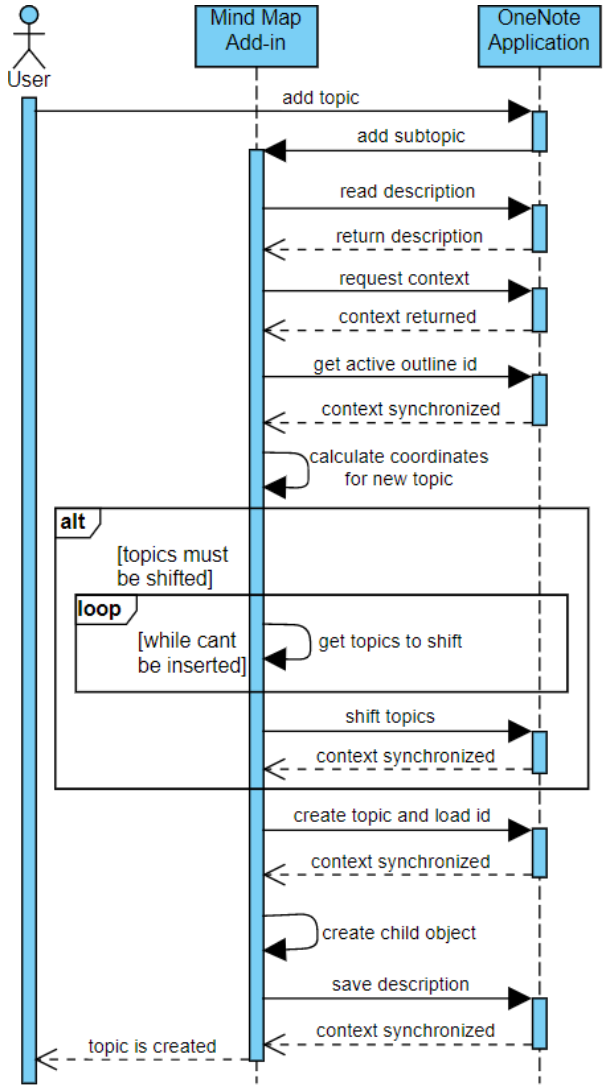
Po vytvoření hlavního tématu myšlenkové mapy dojde ke konverzi proměnných, které ji reprezentují, do JSON řetězce, který bude následně uložen jako popisek obrázku, jenž reprezentuje obrys hlavního tématu. Při jakékoliv následující editaci myšlenkové mapy se tento popisek nejprve načte a na jeho základě se naplní proměnné, které reprezentují existující myšlenkovou mapu. Po provedení požadované operace s myšlenkovou mapou se aktualizované lokální proměnné opět převedou do JSON formátu a uloží se jako popisek obrázku hlavního tématu. Přestože tento přístup neřeší problém dvou existujících stránek doplňku, je velmi užitečný v tom, že umožňuje editovat jednu myšlenkovou mapu více uživatelům. Pokud jeden uživatel povolí přístup jiným uživatelům ke své OneNote stránce a každý z nich bude mít nainstalovaný doplněk pro vytváření myšlenkových map, budou všichni schopni vytvářet a editovat stejnou myšlenkovou mapu, která je uložena na stránce jednoho z nich.

Získání dříve vytvořeného popisu je větší problém než jeho nastavení, protože nevíme, ve kterém obrázku se takový popisek nachází. Musíme tedy projít všechny obrázky ve stránce a najít takový, který má popisek shodující se s jeho strukturou.

Výše zmíněný problém existence dvou identických webových stránek v rámci jednoho doplňku přináší problém, kdy uživatel nebude schopný upravovat vzhled existující myšlenkové mapy. Pokud při provádění funkce pro smazání či vytvoření témat dojde k smazání nebo vytvoření nové úrovně myšlenkové mapy, okno, pomocí kterého lze měnit vlastnosti jednotlivých témat či celé úrovně najednou, tuto změnu nijak nezaregistruje a neposkytne či nezakáže editaci jejích vlastností. To může vyústit k chybě, kdy uživatel bude editovat vzhled úrovně, jež neexistuje, případně nebude schopen editovat novou úroveň, která se již v myšlenkové mapě nachází. Proto má webová stránka doplňku zaregistrovanou událost *DocumentSelectionChanged*, která nastane pokaždé, když dojde ke změně aktivního objektu ve stránce. Pokud uživatel přidá nové téma, má již k dispozici prostředky pro změnu jeho vzhledu. Pro vytvoření další úrovně myšlenkové mapy, tedy přidání dalšího tématu, ale musí kliknout na téma, které již přidal. To vyvolá událost *DocumentSelectionChanged* a dojde k vykonání funkce, která zjistí, že je nutno přidat možnost editace nové úrovně témat. Nově vytvořené téma tak již lze editovat. Tímto způsobem přidávání nových úrovní k editaci vzhledu je zajištěno, že uživatel bude mít vždy možnost nově přidanou úroveň editovat.

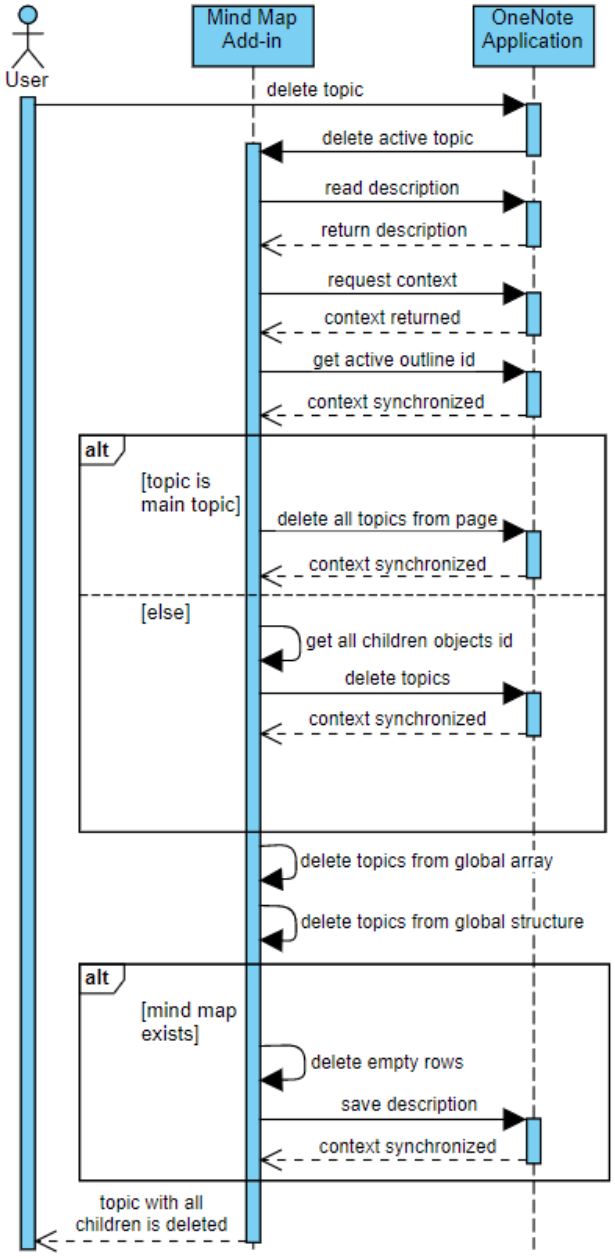
Funkce pro přidání nového tématu, k již existujícímu představuje jednu ze základních funkcí doplňku. Může ji vyvolat uživatel kliknutím na tlačítko „Přidat téma“. Vykonání této funkce v rámci doplňku začíná vyhledáním obrázku, jež obsahuje data ve formátu JSON. Těmito daty bude naplněna globální struktura, jež reprezentuje myšlenkovou mapu. V dalším kroku zjistíme souřadnice obrysu tématu, ke kterému chce uživatel přidat nové téma a vypočítáme z nich nové souřadnice, na které bude přidáno téma. Následně musíme zjistit, jestli je dané místo v myšlenkové mapě již obsazeno. Pokud se na daném místě nachází nějaké téma, je nutno jej posunout tak, aby vznikla volná pozice pro

nové téma. Nyní již můžeme téma přidat do OneNote stránky a načíst jeho *id*, abychom si jej mohli uložit do globální stromové struktury, kterou společně s dalšími informacemi uložíme v JSON formátu jako popisek hlavního tématu. Časová posloupnost vykonávání této funkce, znázorněná sekvenčním diagramem, je vidět na následujícím obrázku. [49]



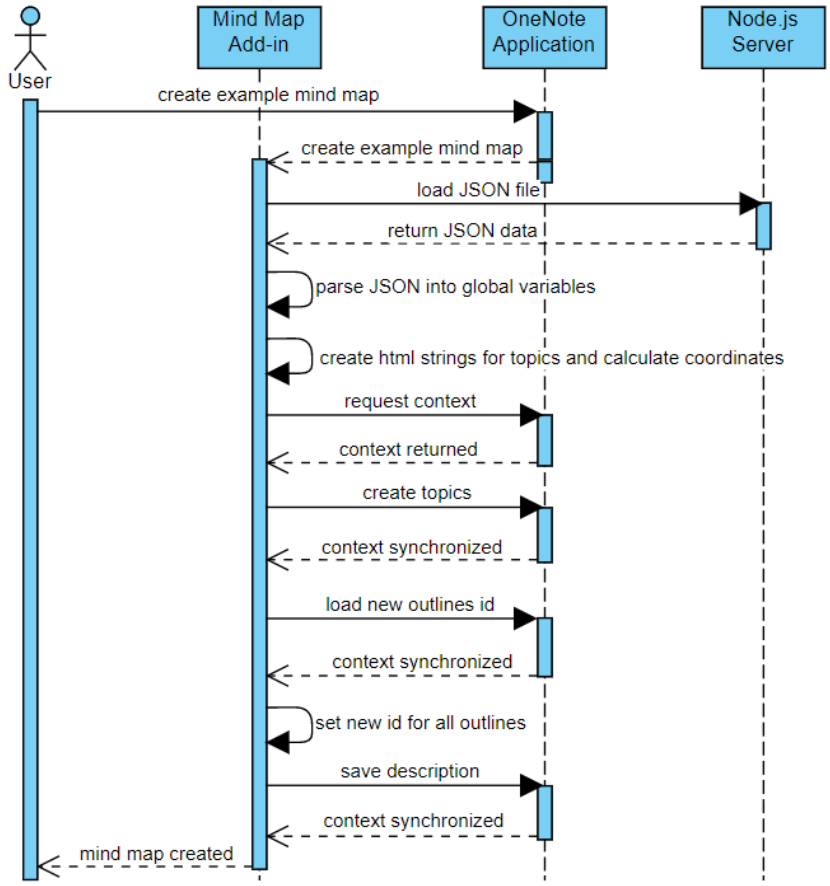
Obrázek 4.7: Sekvenční diagram přidání nového tématu

Další funkcí, bez které se neobejde žádná myšlenková mapa, je funkce pro odstranění uživatelem zvoleného tématu. Začíná velmi podobně jako funkce pro přidání nového tématu, a to načtením dat z popisku obrázku a jejich následným uložením. Dále musíme načíst *id* aktivního tématu, tedy tématu, které chce uživatel smazat. Na jeho základě smažeme dané téma společně se všemi jeho potomky z OneNote stránky. Následně odstraníme dané téma z globálního pole a stromové struktury. Pokud jsme nesmazali celou myšlenkovou mapu, musíme také vyplnit prázdné mezery, které v ní mohly vzniknout smazáním tématu. Posledním krokem je přepsání popisku obrysu hlavního tématu daty, která reprezentují celou myšlenkovou mapu.



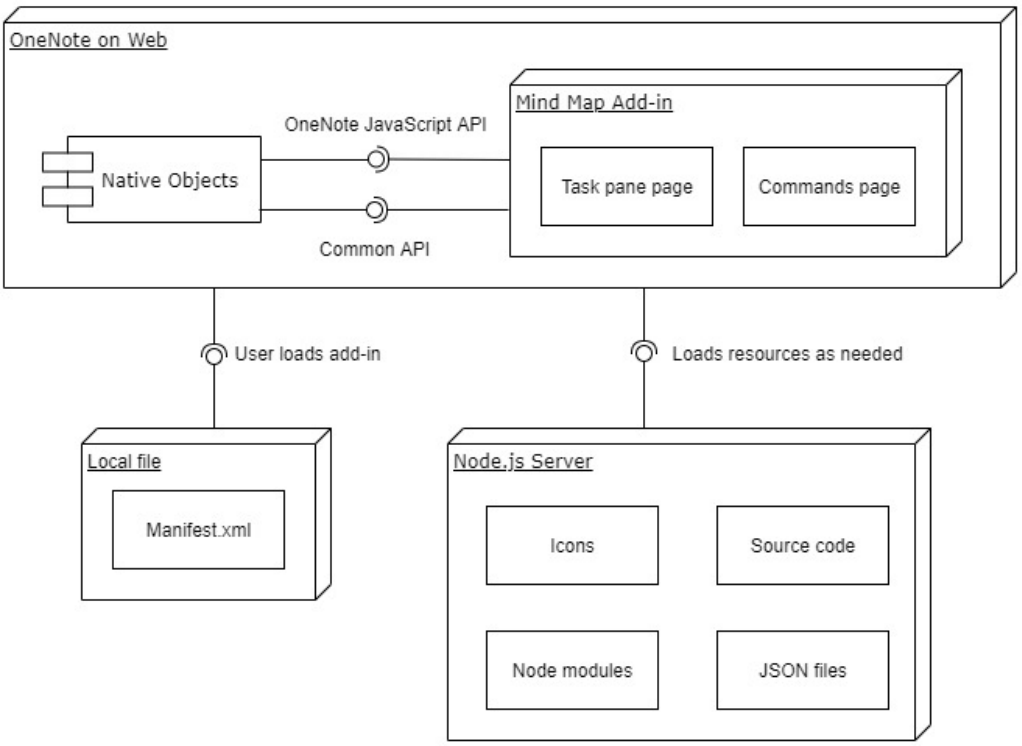
Obrázek 4.8: Sekvenční diagram odstranění tématu

Funkce vytvoření předpřipravené myšlenkové mapy představuje z pohledu uživatele velice užitečnou funkci, jež umožňuje vytvořit vzorovou myšlenkovou mapu v jediném kroku. Nejdříve musíme ze serveru načíst data, která obsahují informace o tom, jak má myšlenková mapa vypadat. Na základě těchto informací vytvoříme HTML řetězce a vypočítáme souřadnice *x* a *y*. Vložíme na ně jednotlivé objekty, ze kterých se myšlenková mapa skládá. Abychom s touto myšlenkovou mapou mohli dále pracovat, musíme načíst *id* jednotlivých objektů, které jsme právě vytvořili a přepsat jimi původní *id*, která jsme načetli ze serveru. Na závěr je potřeba uložit všechna data reprezentující aktuální myšlenkovou mapu jako popisek hlavního tématu. Celý postup vykonávání této funkce je znázorněn sekvenčním diagramem na následujícím obrázku.



Obrázek 4.9: Sekvenční diagram vytvoření předpřipravené myšlenkové mapy

Na následujícím obrázku je vyobrazena architektura doplňku a propojení jednotlivých částí, ze kterých se doplněk skládá.



Obrázek 4.10: Architektura doplňku

4.3 Implementace

Samotná implementace doplňku využívá principy a metodiky popsané v předchozí podkapitole. V následující části budou popsány klíčové konstrukce, které jsou společné pro velkou část funkcí, jež doplněk podporuje.

Většina funkcí začíná tím, že je třeba najít hlavní téma a přečíst jeho popisek, aby se načetla struktura myšlenkové mapy společně s dalšími informacemi, které jsou nezbytné pro jejich správné vykonání.

V rámci tohoto procesu nejdříve projdeme všechny objekty ve stránce a uložíme si ty, které jsou typu *Outline*. Outline obsahuje libovolný počet objektů různých typů, my si ale uložíme pouze obrázky. Pokud jeden z těchto obrázků obsahuje znak *„|“*, je velice pravděpodobné, že obsahuje hledaný popisek, jelikož takový znak odděluje hlášku pro uživatele od relevantních dat. Pokud jsou tato data uložena ve formátu JSON, jedná se s největší pravděpodobností o hledaný popis, který si uložíme a na jeho základě naplníme globální proměnné, jež reprezentují myšlenkovou mapu.

```
async function readDescription() {
  await OneNote.run(async function (context) {
    let page = context.application.getActivePage();
    page.load("contents/items/type");

    await context.sync().then(async function () {
      let paragraphs = [];
      page.contents.items.forEach((item) => {
        if (item.type === "Outline") {
          item.load("outline/paragraphs/items/type");
          paragraphs.push(item.outline.paragraphs);
        }
      });

      await context.sync().then(async function () {
        var images = [];
        paragraphs.forEach((paragraph) => {
          paragraph.items.forEach((item) => {
            if (item.type == "Image") {
              item.image.load("description");
              images.push(item.image);
            }
          });
        });

        await context.sync().then(function () {
          for (let i = 0; i < images.length; i++) {
            let desc = images[i].description;
            if (desc.includes("|")) {
              let string = desc.split("|")[1];
              try {
                var mindMapObject = JSON.parse(string);
              } catch (e) {
                continue;
              }
              parseDescription(mindMapObject);
              return;
            }
          }
        });
      });
    });
  });
}
```

Obrázek 4.11: Získání dat reprezentujících myšlenkovou mapu z popisku obrázku

Součástí přidávání tématu je také vytvoření objektů, jež budou přidány na OneNote stránku. Tento proces již byl částečně popsán v minulé kapitole. K vytvoření takového objektu jsou potřeba souřadnice x a y a řetězec popisující html element, který bude na stránku přidán. Při vytváření textu tématu je jako html element zvolen tag *<p>*, jenž obsahuje atributy, které definují barvu a velikost textu.

```
function createText(text, textsize, textcolor, bgcolor) {
  if (bgcolor === "white" || bgcolor === "#ffffff") {
    var html = `<p style="color: ${textcolor}; font-size: ${textsize}px;">${text}</p>`;
  } else {
    var html = `<p style="background-color: ${bgcolor}; color: ${textcolor}; font-size: ${textsize}px;">${text}</p>`;
  }

  return html;
}
```

Obrázek 4.12: Příklad vytvoření html tagu pro přidání textu do OneNote stránky

Při vytváření obrysu tématu, případně čáry vedoucí od jednoho tématu k druhému, je použit tag *<img>*, přičemž má nastaveny atributy *width*, *height* a *src*. Takový způsob vytváření obrysu tématu, kdy je na dané místo vložen obrázek, je nevýhodný v tom, že jeho vložení do OneNote stránky není okamžité a v závislosti na velikosti obrázku trvá nějakou dobu, než se obrázek zobrazí. Jedná se ale vlastnost aplikace OneNote pro web, kterou není nijak možno ovlivnit.

```
function createImage(width, height, bgcolor, imgcolor, imageType) {
  var canvas = createCanvas(width, height, imageStroke, bgcolor, imgcolor);
  var ctx = canvas.getContext("2d");

  if (imageType === "ellipse") {
    drawEllipse(ctx, width, height, imageStroke);
  } else if (imageType === "rectangle") {
    drawRectangle(ctx, width, height, imageStroke);
  } else if (imageType === "rhombus") {
    drawRhombus(ctx, width, height, imageStroke);
  }

  if (bgcolor !== "white" && bgcolor !== "#ffffff") {
    ctx.fill();
  }
  ctx.stroke();

  var img = canvas.toDataURL("image/png");
  var html = `<img width="${width}" height="${height}" src="${img}"/>`;

  return html;
}
```

Obrázek 4.13: Příklad vytvoření html tagu pro přidání obrysu do OneNote stránky

Postup pro nakreslení šipky spojující dvě téma je velice podobný, jako při kreslení obrysu tématu. Liší se pouze ve výpočtu souřadnic, mezi kterými budou nakresleny čáry reprezentující šipku. Jelikož při vytváření nového tématu víme pouze ve kterém směru a v jakém vertikálním posunutí bude přidáno nové téma, musíme nejprve vypočíst počáteční a cílové souřadnice šipky. Dalším krokem je vypočtení souřadnic, ve kterých budou nakresleny dvě čáry reprezentující hrot šipky. Nejprve pomocí dříve vypočtených souřadnic spočítáme úhel, ve kterém bude vytvořena hlavní čára. K tomu použijeme funkci *Math.atan2*, která vrací úhel v radiánech mezi osou x a čarou představující směr šipky. K tomuto úhlu přičteme, případně odečteme hodnotu π/6 v závislosti na tom, pro kterou šipku počítáme souřadnice. To znamená, že čáry tvořící hrot šipky budou s hlavní čárou šipky svírat úhel 30°. Pokud bychom chtěli, aby mezi nimi byl větší úhel, místo hodnoty π/6 bychom museli použít například hodnotu π/4. Takto vypočtený úhel ale musíme převést na souřadnice, abychom věděli, kam čáry nakreslit. To provedeme funkcemi *Math.cos* pro souřadnice x a *Math.sin* pro souřadnice y. Tyto hodnoty ještě vynásobíme délkou čáry repreztnující hrot a odečteme je od cílových souřadnic šipky. [50] [51] [52]

```
let x1 = direction === "right" ? 0 : width;
let y1 = shift < 0 ? height : 0;
let x2 = direction === "right" ? width : 0;
let y2 = shift < 0 ? 0 : height;

let dx = x2 - x1;
let dy = y2 - y1;
let angle = Math.atan2(dy, dx);
let arrowlength = 10;
let arrow1x = x2 - arrowlength * Math.cos(angle - Math.PI / 6);
let arrow1y = y2 - arrowlength * Math.sin(angle - Math.PI / 6);
let arrow2x = x2 - arrowlength * Math.cos(angle + Math.PI / 6);
let arrow2y = y2 - arrowlength * Math.sin(angle + Math.PI / 6);
```

Obrázek 4.14: Výpočet souřadnic pro nakreslení šipky mezi tématy [53]

Takto vytvořené textové řetězce již je možno jednoduše přidat na stránku zavoláním metody *page.addOutline(x, y, html)*.

Každá funkce, která manipuluje s myšlenkovou mapou, musí být zakončena uložením globálních proměnných ve formě popisku hlavního tématu. Pro jejich uložení je třeba nejdříve získat objekt, který reprezentuje obrázek hlavního tématu. Poté vytvoříme popis ve formátu JSON řetězce a metodou *set* obrázku nastavíme vlastnost *description*. Nakonec je nutno zavolat funkci *sync*, která zajistí, že bude nastavena vlastnost OneNote objektu.

```
async function setDescription(context) {
  let page = context.application.getActivePage();
  let pageContent = page.contents.getItem(topics.idImage);
  let image = pageContent.outline.paragraphs.getItemAt(0).image;

  let description = makeDescription();
  image.set({ description: description });

  return context.sync();
}
```

Obrázek 4.15: Vytvoření popisku obrázku

Jak již bylo zmíněno, doplněk umožňuje zobrazení v českém i anglickém jazyce. Pro změnu jazyka je třeba změnit texty v pásu karet na hodní části obrazovky, v dialogovém okně pro výběr souboru a v podokně na pravé straně obrazovky. To, jaký text se zobrazí v názvu záložky, tlačítka nebo menu je určeno v souboru manifest.xml. Ten má také nastaven výchozí jazyk. Pro všechny tyto texty ale lze určit ekvivalentní text pro jiné jazykové nastavení tak, jak je to znázorněno na následujícím obrázku.

```
<bt:String id="CreateButton.Label" DefaultValue="Hlavní téma">
  <bt:Override Locale="en-us" Value="Main Topic"/>
</bt:String>
```

Obrázek 4.16: Příklad vícejazyčného zobrazení textu

Překlad textů v dialogovém okně a v podokně na pravé straně obrazovky je řešen pomocí speciálního objektu *UIStrings*, který obsahuje verze textů pro jednotlivé jazyky. Pomocí Office API a její vlastnosti *displayLanguage* lze zjistit aktuální jazykové nastavení, na jehož základě získáme texty pro jazyk, ve kterém se zobrazuje aplikace OneNote. Poté je třeba nastavit všechny texty ve stránce na

lokalizovanou hodnotu uvedenou v objektu *UIStrings*. Jediný text, který nelze takto nastavit, je výchozí text elementu *<input type="text">*. Ten určuje prohlížeč na základě jazyka, ve kterém je prohlížeč spuštěn a není možno jej změnit. Řešením takového problému by bylo skrytí tohoto elementu a vytvoření vlastního tlačítka, které by se chovalo stejně jako výše zmíněný html element.

```
UIStrings.CZ = {
  firstButton: "Hlavní téma",
  checkboxLabel: "Nastavit pro celou úroveň",
  textColorLabel: "Barva textu",
  backgroundTextColorLabel: "Barva pozadí textu",
  fontInputLabel: "Velikost textu",
  imageColorLabel: "Barva obrysu",
  backgroundImageColorLabel: "Barva pozadí obrysu",
  imageTypeLabel: "Obrys",
  newLevel: "Úroveň ",
  title: "Nástroj pro vytváření myšlenkových map",
  fileTitle: "Zadej soubor:",
  confirm: "Potvrdit",
  cancel: "Zrušit",
  noFileSelected: "Musíš zvolit jeden soubor",
  wrongFileType: "Špatný typ souboru."
};

UIStrings.getLocaleStrings = function(locale) {
  var text;
  locale = locale.toLowerCase();

  // Get the resource strings that match the language.
  switch (locale) {
    case "en-us":
      text = UIStrings.EN;
      break;
    case "cs-cz":
      text = UIStrings.CZ;
      break;
    default:
      text = UIStrings.EN;
      break;
  }

  return text;
};
```

Obrázek 4.17: Struktura objektu UIStrings, který obsahuje verze textů pro jednotlivé jazyky

4.4 Možnosti rozšíření doplňku

Doplněk je momentálně ve stavu, kdy nabízí celou řadu základních i pokročilých funkcí společně s intuitivním a přívětivým uživatelským rozhraním. Lze jej využít k nakreslení rozmanité a barevné myšlenkové mapy, jejíž témata mohou mít různé tvary. Nicméně jako prakticky každý software, i doplněk pro vytváření myšlenkových map v aplikaci OneNote má mnoho prostoru pro různá vylepšení, která ještě více usnadní vytváření myšlenkových map, nebo přidají nové funkce, jež přispějí k ještě lepší uživatelské zkušenosti.

První funkcí, kterou by dokázal uživatel ocenit, je export myšlenkové mapy ve formě obrázku. Doplněk aktuálně poskytuje pouze export myšlenkové mapy ve formě JSON souboru, který slouží k opětovnému vytvoření myšlenkové mapy po importování tohoto souboru. Funkce pro export myšlenkové mapy by mohla jít spustit tlačítkem, které by bylo umístěno v pásu karet vedle již existujících tlačítek. Při vytváření obrázku, který si uživatel stáhne, můžeme využít již implementované

postupy. Obrysy témat a šipky, jež je spojují, jsou kresleny na plátno, ze kterého je následně vytvořen obrázek, který bude vložen do OneNote stránky. Stejný postup lze použít při exportování myšlenkové mapy ve formě obrázku s tím rozdílem, že nebudeme kreslit pouze jedno téma či šipku, ale celou myšlenkovou mapu, kterou musíme doplnit i texty jednotlivých témat. Takto vytvořený obrázek si uživatel může stáhnout stejným způsobem, jako stahuje JSON soubor reprezentující myšlenkovou mapu.

Další funkce, která by stála za implementaci, je importování myšlenkové mapy, z již existujících aplikací, které umožňují jejich tvorbu. Webová stránka mindup.com poskytuje prostředí, ve kterém je možno myšlenkové mapy vytvářet. Uživatel má možnost myšlenkovou mapu exportovat v několika různých formátech. Doplněk aplikace OneNote by mohl uživateli nabídnout importovat myšlenkovou mapu nejen ve vlastním formátu ve formě JSON souboru, ale také v dalších formátech, které podporují ostatní online prostředí pro tvorbu myšlenkových map.

Doplněk momentálně neumožňuje změnit jak vertikální, tak horizontální vzdálenost mezi tématy, velikost jednotlivých obrysů témat či jejich výchozí text. Všechny tyto vlastnosti jsou reprezentovány pomocí proměnných ve zdrojovém kódu. Stačí pouze umožnit uživateli tyto proměnné editovat buď pomocí tlačítek v záložce karet nebo v podokně na pravé části obrazovky.

Pokud uživatel při vytváření myšlenkové mapy nebo při pokusu o vytvoření druhé myšlenkové mapy v rámci jedné OneNote stránky klikne na tlačítko vytvořit myšlenkovou mapu, dojde k tomu, že v rámci jedné OneNote stránky budou existovat dvě myšlenkové mapy, přičemž uživatel bude moct pracovat pouze s jednou. Aby se zamezilo tomuto nevhodnému chování, měl by být uživatel varován, že se pokouší přidat druhou myšlenkovou mapu do jedné OneNote stránky, ve které může existovat pouze jedna myšlenková mapa. Pokud si uživatel opravdu přeje vytvořit novou myšlenkovou mapu, měla by být ta stará korektně smazána z OneNote stránky a na její místo by se měla přidána nová.

Při vytvoření hlavního tématu se obrys společně s textem vloží na předem definované místo. Přidávání dalších témat do již existující myšlenkové mapy tak může zakrýt již existující poznámky či další uživatelem vytvořený obsah, který není součástí myšlenkové mapy. Pokud by měl uživatel k dispozici čtyři tlačítka, díky kterým by mohl pohybovat celou myšlenkovou mapou najednou ve všech čtyřech směrech, mohl by tím vyřešit problém, kdy by se existující myšlenková mapa překrývala s ostatními objekty ve OneNote.

Kapitola 5

Závěr

Cílem práce je vytvoření uživatelsky přívětivého a funkčně zdatného doplňku aplikace OneNote v rámci které dojde k zefektivnění vytváření myšlenkových map. Myšlenkové mapy jsou vhodným nástrojem pro efektivní záznam mentálních procesů a jsou osvědčeny v plánování nejrůznějších činností pracovního i každodenního života. Tento nově vytvořený doplněk má silný potenciál pro zatraktivnění výuky, což mohou ocenit především pedagogové, ale také žáci a studenti k samostudiu.

V práci byly popsány a vysvětleny, myšlenkové mapy, jejich funkčnost a význam. Snahou bylo uvést čtenáře do problematiky kancelářského balíku Microsoft Office, ve kterém je zahrnuta stěžejní aplikace pro tuto bakalářskou práci, a to Microsoft OneNote. Dále byly specifikovány tři stávající doplňky, které lze použít k vytváření myšlenkových map. Dva z těchto doplňků lze použít pouze pro importování myšlenkové mapy ve formě obrázku, který je nutno nakreslit v externím programu mimo OneNote. Třetí doplněk již umožňuje vytváření myšlenkových map nativně v aplikace OneNote, avšak je dostupný pouze pro OneNote verze 2010, 2013 a 2016. Všechny tyto doplňky jsou zpoplatněny.

Dále byly v práci uvedeny programovací jazyky, jejich konstrukce a technologie, jež byly použity při implementaci doplňku. Pomocí těchto jazyků byl vytvořen nástroj, díky němuž lze myšlenkové mapy vytvářet a upravovat. Jejich tvorba je díky tomuto doplňku uživatelsky přívětivější a není tak časově náročná, jako bez něj.

Poslední část práce byla věnována samotné implementaci. Nejdříve byly specifikovány jednotlivé funkce, které doplněk poskytuje a jejich možnosti použití z pohledu uživatele. Následně byl popsán návrh doplňku společně s jeho architekturou a klíčovými konstrukcemi tvořícími jádro celé aplikace. Nakonec byly zmíněny možnosti, jak doplněk rozšířit a přidat nové funkce. Pro práci byl vyvinut jednoduchý, efektivní a časově nenáročný nástroj k vytváření a úpravě komplexních myšlenkových map. Výstupem práce jsou tedy zdrojové kódy tvořící funkční, konkurenčně schopnou a prakticky použitelnou aplikaci, která je ve formě doplňku pro Microsoft OneNote a jejíž myšlenkovou mapu může upravovat více uživatelů.

Literatura

[1] MYŠLENKOVÉ MAPY. Paměť a učení [online]. [cit. 2021-02-26]. Dostupné z: http://www.pametauceni.cz/clanek-3880-myslenkove-mapy/

[2] Building Office COM add-ins. ADD-IN EXPRESS [online]. [cit. 2021-03-20]. Dostupné z: https://www.add-in-express.com/docs/net-first-com-addin.php

[3] OneNote JavaScript API requirement sets. Microsoft [online]. 24. 8. 2020 [cit. 2021-03-20]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/add-ins/reference/requirement-sets/onenote-api-requirement-sets

[4] Office client application and platform availability for Office Add-ins. Microsoft [online]. 19. 2. 2021 [cit. 2021-03-20]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/add-ins/overview/office-add-in-availability#onenote

[5] URBAN, Petr. OneNote 2016 se tak trochu vrací ze záhrobí. Nabídne tmavý režim: Produkt odstavený na vedlejší koleji se dočkal malého oživení. Jak se změní podpora OneNotu 2016? Cnews.cz [online]. 7. 11. 2019 [cit. 2021-03-20]. Dostupné z: https://www.cnews.cz/onenote-2016-zmena-doby-podpory-office-2019

[6] Microsoft 365. Microsoft [online]. [cit. 2021-02-26]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365?rtc=1

[7] Vyberte si řešení, které vám bude vyhovovat. Microsoft [online]. [cit. 2021-02-24]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365/buy/compare-all-microsoft-365-products

[8] DUFFY, Jill. Microsoft OneNote Review. PCMag [online]. [cit. 2021-02-24]. Dostupné z: https://www.pcmag.com/reviews/microsoft-onenote

[9] CORCORAN, Leonie. How Tony Buzan used mind maps to doodle his way to millions. The Irish Times [online]. 2015, 1. 6. 2015 [cit. 2021-02-24]. Dostupné z: https://www.irishtimes.com/business/how-tony-buzan-used-mind-maps-to-doodle-his-way-to-millions-1.2230977

[10] JAKUBEKOVÁ, Ivana. Mozkové hemisféry. MENTEM [online]. 2014, 8. září 2014 [cit. 2021-02-24]. Dostupné z: https://www.mentem.cz/blog/mozkove-hemisfery/

[11] OneMind for Mac OneNote. In: Office OneNote Gem Add-Ins [online]. [cit. 2021-02-24]. Dostupné z: http://www.onenotegem.com/a/addins/onemind-for-onenote.html

[12] Mind Map for OneNote 9.5.0.67. In: Office OneNote Gem Add-Ins [online]. [cit. 2021-02-24]. Dostupné z: http://www.onenotegem.com/a/addins/mind-map-for-onenote.html

[13] Mind Mapping Diagram. Visual Paradigm [online]. [cit. 2021-02-24]. Dostupné z: https://www.visual-paradigm.com/features/mind-mapping-diagram-and-tools

[14] About JavaScript. MDN Web Docs [online]. c2005-2021 [cit. 2021-02-24]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

[15] JavaScript Let. W3schools [online]. c1999-2021 [cit. 2021-02-24]. Dostupné z: https://www.w3schools.com/js/js_let.asp

[16] JavaScript Arrays. W3schools [online]. c1999-2021 [cit. 2021-02-24]. Dostupné z: https://www.w3schools.com/js/js_arrays.asp

[17] Classes. MDN Web Docs [online]. c2005-2021 [cit. 2021-02-24]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

[18] JavaScript Classes. W3schools [online]. c1999-2021 [cit. 2021-02-24]. Dostupné z: https://www.w3schools.com/js/js_classes.asp

[19] Introducing asynchronous JavaScript. MDN Web Docs [online]. c2005-2021 [cit. 2021-02-24]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing

[20] Making asynchronous programming easier with async and await. MDN Web Docs [online]. c2005-2021 [cit. 2021-02-24]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async_await

[21] HTML: HyperText Markup Language. MDN Web Docs [online]. c2005-2021 [cit. 2021-02-24]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML

[22] MORAES, Frank. HTML For Beginners The Easy Way: Start Learning HTML & CSS Today Read more: https://html.com/#ixzz6nPE9D4jT. HTML.COM [online]. c2015-2020 [cit. 2021-02-24]. Dostupné z: https://html.com/

[23] CSS Introduction. W3schools [online]. c1999-2021 [cit. 2021-02-24]. Dostupné z: https://www.w3schools.com/css/css_intro.asp

[24] JQUery Introduction. W3schools [online]. c1999-2021 [cit. 2021-02-24]. Dostupné z: https://www.w3schools.com/jquery/jquery_intro.asp

[25] JQuery Selectors. W3schools [online]. c1999-2021 [cit. 2021-02-24]. Dostupné z: https://www.w3schools.com/jquery/jquery_selectors.asp

[26] MÁCA, Jindřich. Lekce 1 - Úvod do Node.js. ITnetwork.cz [online]. [cit. 2021-02-24]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs

[27] What is npm? W3schools [online]. c1999-2021 [cit. 2021-02-24]. Dostupné z: https://www.w3schools.com/whatis/whatis_npm.asp

[28] Node.js NPM. W3schools [online]. c1999-2021 [cit. 2021-02-24]. Dostupné z: https://www.w3schools.com/nodejs/nodejs_npm.asp

[29] Introducing JSON. JSON [online]. [cit. 2021-02-24]. Dostupné z: https://www.json.org/json-en.html

[30] CircularJSON. GitHub [online]. [cit. 2021-02-24]. Dostupné z: https://github.com/WebReflection/circular-json

[31] Office Add-ins platform overview. Microsoft [online]. 14. 10. 2020 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/cs-cz/office/dev/add-ins/overview/office-add-ins

[32] OneNote JavaScript API programming overview. Microsoft [online]. 14. 10. 2020 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/add-ins/onenote/onenote-add-ins-programming-overview

[33] Office Add-ins XML manifest. Microsoft [online]. 18. 3. 2020 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/add-ins/develop/add-in-manifests?tabs=tabid-1

[34] Using the application-specific API model. Microsoft [online]. 8. 9. 2020 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/add-ins/develop/application-specific-api-model

[35] Work with OneNote page content. Microsoft [online]. 19. 3. 2019 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/add-ins/onenote/onenote-add-ins-page-content

[36] Develop Office Add-ins. Microsoft [online]. 14. 10. 2020 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/cs-cz/office/dev/add-ins/develop/develop-overview

[37] Common JavaScript API object model. Microsoft [online]. 30. 4. 2020 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/add-ins/develop/office-javascript-api-object-model

[38] OneNote JavaScript API programming overview. Microsoft [online]. 14. 10. 2020 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/add-ins/onenote/onenote-add-ins-programming-overview

[39] Deploy and publish Office Add-ins. Microsoft [online]. 2. 6. 2020 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/cs-cz/office/dev/add-ins/publish/publish

[40] Make your solutions available in Microsoft AppSource and within Office. Microsoft [online]. 14. 1. 2021 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/store/submit-to-appsource-via-partner-center

[41] Store step-by-step submission guide. Microsoft [online]. 14. 1. 2021 [cit. 2021-02-24]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/store/add-in-submission-guide

[42] Pfleeger, Shari Lawrence, and Joanne M. Atlee. 2009. Software Engineering: Theory and Practice: Prentice Hall, ISBN 0136061699

[43] Pressman, Roger S. 2010. Software Engineering : A Practitioner's Approach. 7th ed. New York: McGraw-Hill Higher Education, ISBN 9780073375977

[44] Sommerville, Ian. 2010. Software Engineering. 9th ed, International Computer Science Series. Harlow: Addison-Wesley, ISBN 978-0137035151

[45] Helpers and tips for npm run scripts. Michael Kühnel [online]. 22. 3. 2018 [cit. 2021-03-12]. Dostupné z: https://michael-kuehnel.de/tooling/2018/03/22/helpers-and-tips-for-npm-run-scripts.html

[46] What is Class Diagram? Visual Paradigm [online]. [cit. 2021-03-12]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

[47] Add-in commands for Excel, PowerPoint, and Word. Microsoft [online]. 29. 1. 2021 [cit. 2021-03-01]. Dostupné z: https://docs.microsoft.com/en-us/office/dev/add-ins/design/add-in-commands

[48] What is Use Case Diagram? Visual Paradigm [online]. [cit. 2021-03-02]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

[49] What is Sequence Diagram? Visual Paradigm [online]. [cit. 2021-03-02]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

[50] Math.atan2(). MDN Web Docs [online]. 9. 3. 2021 [cit. 2021-03-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/atan2

[51] Math.cos(). MDN Web Docs [online]. 19. 2. 2021 [cit. 2021-03-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/cos

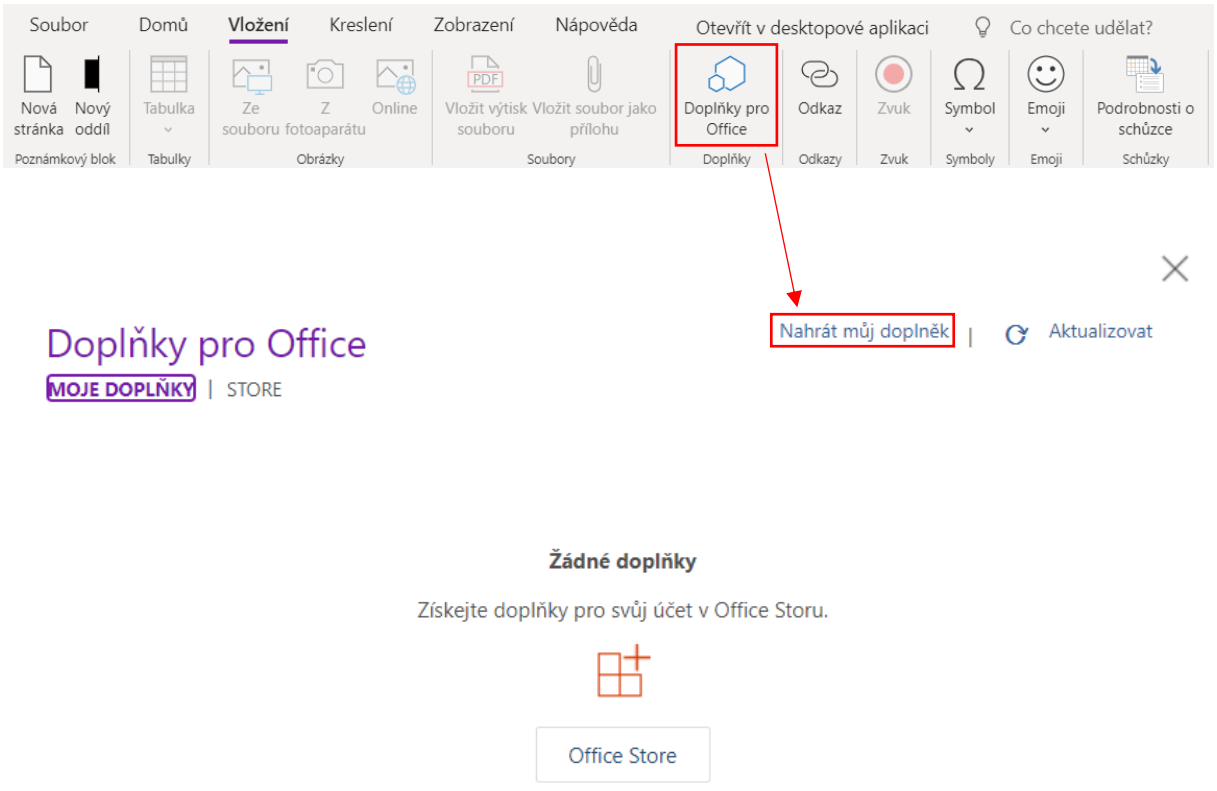
[52] Math.sin(). MDN Web Docs [online]. 19. 2. 2021 [cit. 2021-03-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/sin

[53] Draw arrow on canvas tag. Stack Overflow [online]. 21. 7. 2019 [cit. 2021-03-13]. Dostupné z: https://stackoverflow.com/questions/808826/draw-arrow-on-canvas-tag

Příloha A

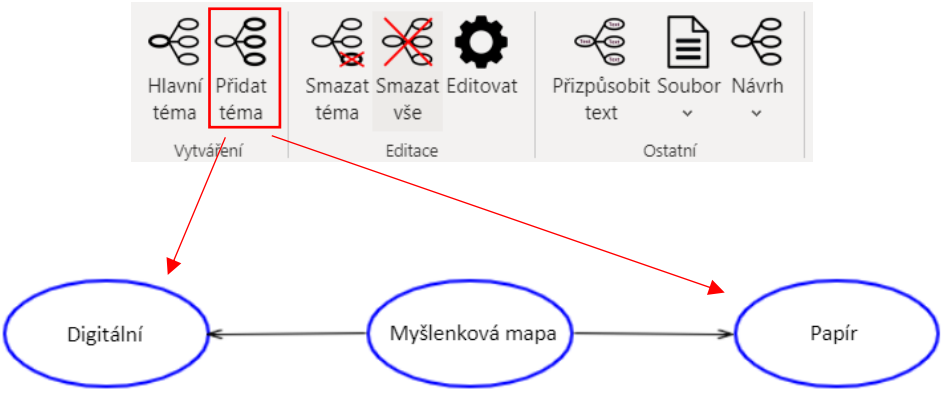
Příklad použití doplňku vytvořeného pro tuto bakalářskou práci:

1. Prvním krokem je instalace Node.js serveru. Instalátor lze stáhnout ze stránky https://nodejs.org/en/download/.
2. Následně je třeba stáhnout moduly, které jsou nutné pro běh serveru. V kořenovém adresáři projektu otevřeme příkazový řádek s právy administrátora a v něm zadáme příkaz npm install.
3. Nyní je potřeba zapnout server. V již otevřeném příkazovém řádku zadáme příkaz npm run start:web a odsouhlasíme případné volby či potvrdíme důvěryhodnost instalovaného certifikátu.
4. Dále provedeme vložení doplňku do aplikace OneNote pro web, která je dostupná na stránce www.onenote.com. Pomocí ikony „Doplňky pro Office“, která se nachází v záložce „Vložení“ otevřeme okno pro výběr doplňků. Zvolíme možnost „Nahrát můj doplněk“, vybereme soubor manifest.xml a tlačítkem „Nahrát“ volbu potvrdíme.

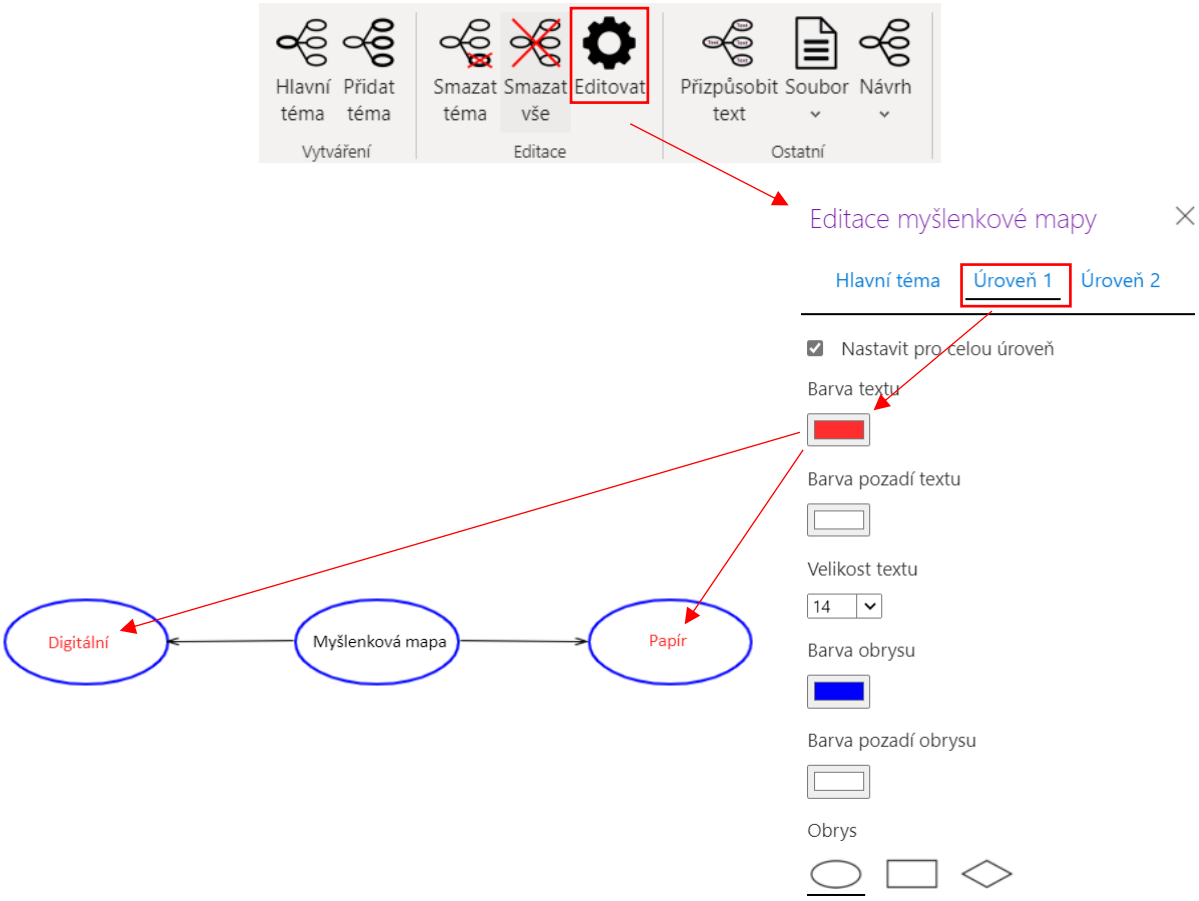


5. Po dokončení předchozího kroku dojde k vytvoření nové záložky nazvané „Myšlenková mapa“, která poskytuje rozhraní pro vytváření myšlenkové mapy. Zvolením tlačítka „Hlavní téma“, které se v této záložce nachází, bude vytvořeno hlavní téma s výchozím textem, který lze následně upravit. Tlačítko „Přizpůsobit text“ tento text posune tak, aby se nacházel uprostřed daného tématu.

6. Dalším krokem je vytváření jednotlivých témat, která dohromady tvoří myšlenkovou mapu. Aby doplněk věděl, se kterým tématem má nově přidané téma spojit, musí uživatel kliknout buď na obrys daného tématu, nebo na jeho text. Následně již může kliknout na tlačítko „Přidat téma“, čímž vytvoří podtéma k právě zvolenému tématu. Poté již stačí změnit výchozí text a tlačítkem „Přizpůsobit text“ jej vycentrovat. Pro smazání tématu a všech jeho podtémat musíme kliknout na text, případně obrys daného tématu a zvolit tlačítko „Smazat téma“.



7. Nakonec, když jsou všechna témata v myšlenkové mapě vytvořena, můžeme upravit jejich vzhled. K tomuto účelu slouží tlačítko „Editovat“, které otevře podokno pro editaci vzhledu myšlenkové mapy. V něm vybereme úroveň myšlenkové mapy, kterou chceme editovat a změníme barvu u jednoho ze vstupních polí. Pro změnu vzhledu jediného tématu nezávisle na tom, do které úrovně patří, musíme odškrtnout zaškrtávací políčko, vybrat téma, které chceme upravit a změnit jedno ze vstupních polí.



8. Proces vytváření myšlenkové mapy lze urychlit vytvořením předpřipravené myšlenkové mapy. Kliknutím na tlačítko „Návrh“ se otevře menu, ve kterém si uživatel může vybrat jednu ze tří velikostí myšlenkové mapy, která bude následně vytvořena.